Generador de gestores de contenido web en Java

Memòria del projecte

d'Enginyeria Tècnica en

Informàtica de Sistemes

realitzat per

Joaquín Durán Álvaro

i dirigit per

Mercedes Narciso Farias

Escola d'Enginyeria

Sabadell, juny de 2011

La sotasignat, Mercedes Narciso Farias,

professora de l'Escola d'Enginyeria de la UAB,

CERTIFICA:

Que el treball al que correspon la present memòria ha estat realitzat sota la seva direcció per

Joaquín Durán Álvaro

I per a que consti firma la present.
Sabadell, juny de 2011

Signat: Mercedes Narciso Farias

Generador de gestores de contenido web en Java

FULL DE RESUM – PROJECTE FI DE CARRERA DE L'ESCOLA D'ENGINYERIA

Títol del projecte: Generador de gestores de contenido web en Java	
Autor: Joaquín Durán Álvaro	**Data:** juny de 2011
Tutora: Mercedes Narciso Farias	
Titulació: Enginyeria Tècnica en Informàtica de Sistemes	

Paraules clau (mínim 3)

- Català: Java, Framework, JSP, Struts, Mysql.
- Castellà: Java, Framework, JSP, Struts, Mysql.
- Anglès: Java, Framework, JSP, Struts, Mysql.

Resum del projecte (extensió màxima 100 paraules)

- **Català:** Aquest projecte consisteix en el desenvolupament d'una aplicació web programada en Java, que permet crear diferents tipus de webs, donant la opció de definir i administrar tant les estructures com els continguts dels apartats. Altres opcions que permet gestionar són els tipus d'usuaris que podran accedir i el disseny gràfic de cada web generada. S'aconsegueix així crear webs sense necessitat de programar.

  Mitjançant el desenvolupament del projecte es pretén aprendre i utilitzar el framewok conegut com anomenat Struts que ofereix una plataforma sobre la qual es desenvolupa l'aplicació permetent utilitzar una sèrie de llibreries ja programades i treballar utilitzant una metodologia o patró de disseny.

- **Castellà**: Este proyecto consiste en el desarrollo de una aplicación web programada en Java, que permite crear diferentes tipos de webs, dando la opción de definir y administrar tanto las estructuras como los contenidos de los apartados. Otras opciones que permite gestionar son los tipos de usuarios que podrán acceder y el diseño gráfico de cada web generada. Se consigue así crear webs sin necesidad de programar.

  Mediante el desarrollo del proyecto se pretende aprender y utilizar el *framewok* conocido como Struts que ofrece una plataforma sobre la que se desarrolla la aplicación permitiendo utilizar una serie de librerías ya programadas y trabajar utilizando una metodología o patrón de diseño.

  Anglès: This project is developing a web application programmed in Java, which lets you create different types of sites, giving the option to define and manage both structures and contents of the sections. Other options for managing the types of users can access and the graphic design of each site generated. This makes for creating websites without programming.

  Through the development of the project is to learn and use framewok called Struts that provides a platform on which the application is developed allowing the use of a number of libraries already scheduled and work through an methodology or design pattern.

Generador de gestores de contenido web en Java

Índice de contenidos

1. Introducción

1.1. Descripción del contexto

Un sistema de gestión de contenidos también conocido como CMS *(Content Management System)* es un programa que ofrece una interfaz de edición gráfica para la creación y administración de *contenidos*[1], mayoritariamente de webs. Estos contenidos están almacenados en una base de datos, separándolos así del código de la web, que no varía como por ejemplo del que representa el aspecto gráfico, dando la posibilidad al usuario de poder gestionar teles contenidos fácilmente sin tener conocimientos de programación, y al desarrollador la posibilidad de poder cambiar el diseño de la web sin que altere el contenido. Esto tiene como consecuencia el abaratamiento de los costes de desarrollo, ya que cada vez que se realiza un cambio no se hace necesario tener que modificar el código de la web, tarea que en principio se realiza una sola vez al inicio de la instalación.

Para la realización del proyecto se desarrollará un CMS utilizando el entorno de programación o *framework* llamado Struts, que consiste en un conjunto de librerías y código ya programado que forman una plataforma estandarizada ofreciendo una base sobre la que se añadirá el código de la aplicación que se quiera desarrollar, obligando a seguir una metodología de trabajo o patrón de diseño con una serie de soluciones que se podrán aprovechar. Su finalidad es la de facilitar el desarrollo de software permitiendo a los programadores invertir más tiempo identificando requerimientos de software que tratando con los detalles de la programación dedicados a aspectos muy específicos y sin con menor relevancia para la funcionalidad global de la aplicación.

Otra de las ventajas de seguir una metodología de trabajo o patrón de diseño es que consigue que el código de los desarrolladores mejore su estructura aumentando la practicidad y reduciendo las posibilidades de error.

A la hora de realizar una web que necesite interaccionar con una base de datos, se puede utilizar un lenguaje de programación interpretado, lo que significa que se ejecuta en el servidor donde está alojada la web y no en el ordenador desde el que accede el usuario. Dos de los lenguajes de programación interpretados más utilizados son el PHP (*Hypertext Pre-Processor*) y el Java.

En la realización del proyecto se ha optado por el lenguaje Java porque permite realizar aplicaciones más potentes sobre todo en el ámbito de comunicación de redes, por ejemplo realizar un programa que se ejecute desde escritorio de nuestro ordenador y interactúe con la web.

[1] *Contenidos*: en el entorno de la programación web cuando se utiliza este término, se hace referencia a los datos que se enseñan en una página web sobre una plantilla que al cambiar a una página de su misma categoría se sigue utilizando la misma plantilla cambiando únicamente los datos de contenidos, que normalmente son leídos de una base de datos. De esta manera se consigue que programando una sola página se pueda mostrar muchas diferentes con la misma estructura y con contenidos diferentes.

1.2. Objetivos

Hoy en día en la programación de web es muy importante la utilización de frameworks y CMS. Por eso como objetivo principal se va a utilizar la realización del proyecto para obtener conocimientos sobre unos de los frameworks más utilizados del mercado laboral, además también se va a crear un CMS que ofrezca las funcionalidades que más se adapten a los requisitos que se puedan demandar en la realización de webs. El lenguaje de programación escogido es el Java porque en comparación con el PHP ofrece un abanico más amplio de posibilidades a la hora de programar.

Con la realización del proyecto se pretende desarrollar una herramienta que sirva para crear la mayoría de webs que se demanden sin tener que llegar programarlas, por este motivo y para que esto se cumpla la aplicación se diseñará de forma que en el futuro se puedan incorporar y administrar nuevas funcionalidades de forma sencilla, consiguiendo que cada vez sea capaz de crear un abanico de webs más amplio.

Como resultado del desarrollo del CMS se pretende que sea capaz de gestionar dos tipos de webs diferentes, en el primer caso una web pública de promoción de un pueblo de Burgos y en el segundo caso una agenda virtual de acceso restringido de un alumno universitario.

1.3. Contenido de la memoria

En capítulo **2. Estudio de viabilidad** se realiza la planificación prevista para el desarrollo del proyecto donde se comprobará si resulta viable su realización, explicando la situación actual del mercado laboral, los objetivos a cumplir, los costes y riesgos.

En capítulo **3. Fundamentos teóricos** se presenta la herramienta que se desea aprender y utilizar para la realización del proyecto.

En los capítulos **4. Análisis de requerimientos**, **5. Desarrollo** y **6. Implementación** se presentan las distintas fases de realización de la aplicación que se desarrolla en este proyecto.

En el capítulo **7. Ejemplo de webs creadas** se utilizará la aplicación desarrollada para definir y administrar dos tipos de webs diferentes.

En el capítulo **8. Conclusiones** se muestran las conclusiones obtenidas de la realización de este proyecto y las ampliaciones que se pueden realizar para mejorar el resultado del mismo.

2. Estudio de viabilidad

En este capítulo se analiza si es viable realizar el proyecto, realizando un estudio de los medios necesarios para desarrollarlo tanto económicos como técnicos.

2.1. Situación actual del mercado laboral

Según los recursos que poseen y los requerimientos de sus clientes cada empresa opta por trabajar utilizando una herramienta a la hora de desarrollar una aplicación, ya sea programándola a medida, utilizando frameworks o aprovechando un CMS. Cada una de estas opciones tiene sus ventajas y desventajas.

Las empresas que realizan un proyecto programándolo exclusivamente a medida suelen poseer una gran biblioteca de programación re-aprovechable que han desarrollado a lo largo de su trayectoria, y además las características del tipo de proyecto que realizan suelen ser muy específicas haciendo muy difícil adaptar un CMS para su desarrollo, teniendo que programar la aplicación a su medida. La ventaja de esta opción es que la empresa conoce la programación de sus aplicaciones al 100% pero la gran desventaja que tiene es que pierden mucho tiempo en programarlas ya que no han tenido la ocasión de aprovechar funcionalidades ya programadas por otras personas, como sería en el caso de utilizar un framework o un CMS.

Para las empresas, a la hora de trabajar sería de utilidad utilizar un framework, ya que además de ahorrarse tener que programar muchas funcionalidades resueltas por el framework y de poca relevancia para la funcionalidad general de la aplicación, como por ejemplo, validar un formulario antes de ser enviado, también obliga a seguir un patrón de trabajo que hará que el proyecto siga unos estándares de buenos hábitos, consiguiendo una buena organización y minimizando los errores. Como desventaja, esta elección tiene un aumento de la dificultad respecto a la programación a medida, ya que además de conocer el lenguaje de programación utilizado también hay que conocer los protocolos y reglas a seguir del framework en sí.

Por último, si los requisitos de los clientes son lo suficientemente generales como para no necesitar un exceso de programación a medida y se adaptan a los comportamientos habituales de los CMS, estos serian la mejor opción, ya que ofrecen una serie de funcionalidades a un alto nivel de acabado que pueden satisfacer muchos de los requisitos más comunes. Como desventaja tiene que el desarrollador pierde una gran autonomía sobre el código, dependiendo totalmente de las funcionalidades que ofrece el CMS, que no siempre son todo lo satisfactorias que se quisiera. Esto provoca que a la hora de diseñar un proyecto haya que adaptarse al máximo a la manera de trabajar del CMS pudiendo quedar limitada la practicidad de las funcionalidades de la aplicación, a cambio de un abaratamiento de costes al reducir el tiempo de programación. Otro tema a tener en cuenta es que la gran mayoría están programados en PHP, siendo pocos los CMS programados en Java. Por último también hay que tener en cuenta que son pocos los CMS que permiten crear una web sin llegar a programar nada, ya que normalmente hay que modificar parte del código aunque sea para aspectos del diseño.

2.2. Perfil del usuario

La aplicación a desarrollar está pensada para que la utilicen tres grupos principales de usuarios:

Los usuarios administradores son los encargados de crear la estructura inicial de la web definiendo los tipos de contenidos, los usuarios que podrán acceder a la web y sus respectivas restricciones.

Un segundo grupo de usuario son los editores que se encargarán de introducir el contenido de la web. Dentro de la zona de administración solamente podrán acceder a la sección de la edición del contenido, no pudiendo acceder a ninguna sección donde se pueda cambiar la configuración de la web.

Por último están los usuarios que accederán a los apartados creados de la web, dependiendo de las restricciones que tenga cada usuario solamente podrán acceder a las zonas totalmente públicas o a alguna sección privada.

2.3. Objetivos

A continuación se presentan los objetivos que se pretenden conseguir con la realización del proyecto:

- Adquirir conocimientos sobre el framework llamado Struts y utilizarlo para realizar la aplicación del proyecto.
- Desarrollar de un CMS que permita crear y mantener diferentes tipos de webs sin tener que llegar a programarlas.
- Conseguir que la aplicación disponga de una interfaz gráfica intuitiva y fácil de utilizar.
- Realizar dos tipos de webs diferentes como ejemplo de uso del CMS desarrollado.
- Diseño e implementación de una base de datos que permita gestionar la información de la aplicación.
- Ofrecer un sistema de seguridad robusto que solamente permita acceder a las áreas restringidas a los usuarios que tengan permiso para ello.
- Realizar una aplicación que no sobrecargue los recursos del hardware sobre el que se ejecutará.
- Programar el CMS con la idea de que sea fácil incorporar futuras nuevas funcionalidades, consiguiendo que cada vez se puedan hacer webs más completas.

2.4. Sistema a realizar

2.4.1. Descripción

Se pretende realizar un sistema que ofrezca una interfaz gráfica de acceso restringido desde la cual se pueda definir la estructura, introducir los contenidos y diseñar el aspecto gráfico de diferentes tipos de webs con diferentes tipos de contenidos.

Para la gestión de los datos que utilizará la aplicación se diseñará e implementará una base de datos que sea capaz de almacenar todas las definiciones creadas desde la aplicación y que posteriormente sean traducidas a la web que utilizará el usuario final.

Para poder publicar en internet un entorno programado con el lenguaje Java se necesitará instalar un servidor web.

2.4.2. Método de desarrollo

Primero se realizará un estudio sobre el framework Struts que es la herramienta utilizada para desarrollar la aplicación. Posteriormente se efectuará el análisis diseño e implementación de la aplicación, teniendo en cuenta que se implementará siguiendo el patrón de arquitectura de software MVC (Modelo Vista Controlador), este patrón lo impone el framework y consiste en separar la programación en tres capas diferentes:

Capa Controlador: Se encarga de responder a los eventos que llegan a la aplicación generalmente acciones del usuario, clasificarlos, ejecutar las funcionalidades adecuadas de la capa Modelo que demanda el evento y enviar el resultado a la capa Vista. Esta capa ya dispone de una biblioteca de funcionalidades disponibles para el desarrollador, siendo mínimo el trabajo que se realiza sobre ella.

Capa Modelo: Esta capa contiene la mayoría de la programación que se tiene que desarrollar encargándose básicamente de obtener los datos solicitados de la base de datos y formatearlos adecuadamente para devolver el resultado adecuado.

Capa Vista: La función de esta capa es la de mostrar los resultados finales al usuario. Como en la capa Controlador dispone de un abanico de funcionalidades ya realizadas las cuales minimizan el trabajo de su programación.

Basando el esfuerzo sobre la capa Modelo además de ahorrar tiempo de programación en aspectos ya desarrollados en las otras capas, se consigue un desarrollo general mucho más estructurado y bien definido ganando tiempo en el desarrollo y una reducción de posibles errores. Esta separación que concentra la mayoría del código programado en la capa modelo, también permite que en un futuro se tenga la opción de utilizar un framework diferente más actualizado para el momento o que simplemente lo demande el cliente siendo mínimas las modificaciones a realizar, únicamente desplazando la capa Modelo de un framework a otro.

2.4.3. Recursos

Recursos de hardware

Tanto para el desarrollo del proyecto, como para la posterior realización de pruebas de validación del correcto funcionamiento se utilizará un ordenador portátil con las siguientes características:

Procesador: *Intel Core i5 CPU, M450 2.40GHz.*
Memoria RAM: *4GB.*
Disco duro: *500GB.*
Sistema operativo: *Windows 7 Home Premium.*

Para el alojamiento del CMS se utilizara un ordenador que realice la función de servidor con las siguientes características:

Procesador: *AMD 230, 1.60GHz.*
Memoria RAM: *1GB.*
Disco duro: *250GB.*
Sistema operativo: *La distribución Linux Ubuntu.*

Recursos de software

Para el diseño, análisis y desarrollo del proyecto se utilizarán los siguientes programas:

Servidor de Bases de datos: Mysql.
Editor Java: Netbeans IDE 6.9.1.
Servidor de la aplicación Java: Tomcat.
Editor de Base de datos: Navicat Lite.
Editor de control y seguimiento de proyectos: Microsoft Office Project.
Editor de diagramas UML: Start UML.
Editores de texto: Microsoft Word 2007 y Open Office Writer.
Navegadores de páginas web: Mozilla Firefox, Internet Explorer 6, Internet Explorer 7, Internet Explorer 8, Opera y Chrome.
Plugins del Firefox que ayudan a analizar mostrando la codificación y funcionamiento de las páginas webs: Web Developer, HTML Validator y Firebugs.

Para el alojamiento del CMS se necesitará que el servidor contenga los siguientes programas:

Servidor de Bases de datos: Mysql.
Servidor de la aplicación Java: Tomcat.
Servidor de transferencias de ficheros: Servidor Ftp o consola web.
Servidor de DNS: servidor DNS con IP fija o IP variable con re-direccionamiento de las DNS.

2.4.4. Análisis de coste

Costes de Hardware

Recurso	Coste
Ordenador portátil utilizado para el desarrollo de la aplicación	650 €
Ordenador utilizado para alojar la aplicación como servidor	450 €
Total	**1100 €**

Costes de Software

No se calcula un coste adicional ya que el software utilizado es gratuito o ya venían instalados en la compra de los ordenadores utilizados.

Costes de Recursos Humanos

Recurso	Iniciales	Trabajo	Coste	Coste Total
Director del proyecto	DP	100 h	0 €/h	0 €
Jefe de proyecto	JP	30 h	50 €/h	1500 €
Analista	A	50 h	40 €/h	2000 €
Programador	P	130 h	30 €/h	3900 €
Técnico de pruebas	TP	25 h	20 €/h	500 €
Técnico de sistemas	TS	20 h	30 €/h	600 €
Total		**355 h**	**-**	**8500 €**

Costes totales

Recurso	Coste
Costes de Hardware	1100 €
Costes de Software	0 €
Costes de Recursos Humanos	8500 €
Total	**9600 €**

2.4.5. Evaluación de riesgos

Evaluación de riesgos relacionados con el estudio de viabilidad

- Haber elaborado un cálculo de costes demasiado ajustado provocando que no se consiga acabar el proyecto con los recursos y el tiempo estimado, lo que provocaría una pérdida económica o una pérdida de calidad del proyecto.

Evaluación de riesgos relacionados con la formación de nuevas tecnologías

- No alcanzar la previsión de formación que se pretende sobre el framework Struts que se utilizará para desarrollar la aplicación, lo que provocaría utilizar un sistema de programación más al alcance de la mano, como por ejemplo programar el CMS directamente en PHP.

Evaluación de riesgos relacionados con el análisis y desarrollo de la aplicación

- No conseguir definir un entorno de usuario adecuado que sea lo suficientemente práctico perjudicando la usabilidad de la herramienta.
- Falta de una implementación de medidas de seguridad adecuada que pueda provocar una pérdida de información.
- No conseguir una estructura de programación modular que facilite la fácil incorporación de nuevas funcionalidades en un futuro.

Evaluación de riesgos relacionados con la implementación

- No realizar correctamente la fase de pruebas que permita localizar errores y mejoras en el acabado final de la aplicación.
- No conseguir instalar la aplicación desarrollada en el entorno adecuado que ofrezca los mejores servicios para su correcto funcionamiento.
- Abandono del proyecto antes de su finalización a causa de factores externos habiendo invertido un tiempo sin conseguir una aportación económica, de conocimientos o una herramienta re-aprovechable.

2.4.6. Alternativas

Programación a medida

Esta alternativa aumentaría los costes al invertir demasiado tiempo en la programación de funcionalidades dedicadas a aspectos muy específicos y con menor relevancia para la funcionalidad global de la aplicación que ya ofrecen los frameworks resultas y mejoradas con el paso del tiempo.

Frameworks PHP: CakePHP, Symfony, Zend

La metodología de de estos frameworks es la misma que la de Struts pero con la gran diferencia que están programados en PHP, y se prefiere optar por un entorno de programación en Java ya que se pretender llegar a crear funcionalidades difíciles de conseguir con PHP como por ejemplo conexiones de red en tiempo real para poder crear un Chat.

Framework Java: Spring

Esta opción es la más parecida a la que se ha elegido ya que utiliza una metodología muy similar sobre el mismo lenguaje de programación. Se ha optado por el framework Struts ya que se dispone de más facilidad a la hora de conseguir documentación sobre la estructura.

CMS's eZ Publish, Drupal, Joomla, Magento

La opción de utilizar un CMS está descartada ya que además de ser difícil encontrar uno desarrollado en Java que satisfaga las necesidades planteadas, también ofrecen una serie de dificultades que hacen que se descarten:

- Un CMS está muy bien si los requisitos coinciden con las funcionalidades que ofrece, pero si se quiere alterar estas funcionalidades para que coincidan con los requisitos la dificultad se multiplica exponencialmente.
- Suelen ofrecer un exceso de utilidades no necesarias que provocan una sobrecarga de los recursos de la memoria del hardware ralentizando el tiempo de ejecución de la aplicación.
- En algunos casos aunque sean productos gratuitos, cuando se quiere una documentación más elaborada o conseguir una actualización superior del CMS básico dejan de serlo.

2.5. Planificación

A continuación Figura 2.1 se detalla la plantificación definida para la realización del proyecto, dividida en 7 fases principales, se calcula que la plantilla que forman los recursos humanos van a trabajar 2 horas diarias, sin contar los fines de semana ni periodos vacacionales comprendidos entre los periodos de las fechas 25/11/09 - 8/12/09, 21/12/09-8/01/10 y 2/04/10-6/04/10.

Id	Nombre de tarea	Duración	Comienzo	Fin	Predecesora	Iniciales del
1	Inicio del proyecto: asignación y matriculación del proyecto	1 hora	lun 16/11/09	lun 16/11/09		DP;JP
2	**Planificación.**	**5 días**	**lun 16/11/09**	**lun 23/11/09**		
3	Estudio de viabilidad	10 horas	lun 16/11/09	lun 23/11/09	1	JP
4	**Investigación de los fundamentos teóricos**	**30 días**	**lun 23/11/09**	**mié 03/02/10**		
5	Investigación de entornos de programación web	4 horas	lun 23/11/09	mar 01/12/09	3	A;P
6	Investigación de Struts	40 horas	mar 01/12/09	vie 22/01/10	5	A;P
7	Investigación de entornos de servidor Linux	8 horas	vie 22/01/10	jue 28/01/10	6	TS
8	Redacción de la documentación de fundamentos teóricos	8 horas	jue 28/01/10	mié 03/02/10	7	TS
9	**Análisis de la aplicación**	**8,5 días**	**mié 03/02/10**	**lun 15/02/10**		
10	Análisis de requisitos (diagrama de casos de uso)	4 horas	mié 03/02/10	vie 05/02/10	8	A
11	Análisis de datos (diagrama de clases)	4 horas	vie 05/02/10	mar 09/02/10	10	A
12	Análisis de seguridad	4 horas	mar 09/02/10	jue 11/02/10	11	A
13	Documentación del análisis	4 horas	jue 11/02/10	lun 15/02/10	12	A
14	Aprobación del análisis (Punt de control).	1 hora	lun 15/02/10	lun 15/02/10	13	A;JP;DP
15	**Diseño de la aplicación**	**14,5 días**	**mar 16/02/10**	**lun 08/03/10**		
16	Diseño de la base de datos	5 horas	mar 16/02/10	jue 18/02/10	14	A;P
17	Diseño funcionalidad (diagrama de secuencia)	5 horas	jue 18/02/10	lun 22/02/10	16	A;P
18	Diseño de la interfaz (esquemas de las pantallas)	5 horas	mar 23/02/10	jue 25/02/10	17	A;P
19	Diseño de la batería de pruebas (test)	10 horas	jue 25/02/10	jue 04/03/10	18	A;P;TP
20	Documentación del diseño.	3 horas	jue 04/03/10	vie 05/03/10	19	A
21	Aprobación del diseño (Punto de control).	1 hora	lun 08/03/10	lun 08/03/10	20	DP;A;JP
22	**Desarrollo de la aplicación.**	**50 días**	**lun 08/03/10**	**jue 20/05/10**		
23	Preparación del entorno de desarrollo.	5 horas	lun 08/03/10	mié 10/03/10	21	P
24	Configuración de la base de datos.	5 horas	jue 11/03/10	lun 15/03/10	23	P
25	Programación de la aplicación	80 horas	lun 15/03/10	jue 13/05/10	24	P
26	Desarrollo de la interfaz de usuarios.	10 horas	jue 13/05/10	jue 20/05/10	25	P
27	**Test y pruebas.**	**7,5 días**	**jue 20/05/10**	**lun 31/05/10**		
28	Realización de la batería de pruebas	10 horas	jue 20/05/10	jue 27/05/10	26	TP
29	Documentación de desarrollo y test.	4 horas	jue 27/05/10	lun 31/05/10	28	TP
30	Aprobación del desarrollo i pruebas (Punto de control).	1 hora	lun 31/05/10	lun 31/05/10	29	JP;A;P;DP
31	**Implantación.**	**25 días**	**mar 01/06/10**	**lun 05/07/10**		
32	Utilización de la aplicación para generar webs de ejemplo	5 horas	mar 01/06/10	jue 03/06/10	30	A;P
33	Baterias de pruebas sobre las web de ejemplo	40 horas	jue 03/06/10	jue 01/07/10	32	TP
34	Formación de usuarios.	5 horas	jue 01/07/10	lun 05/07/10	33	A
35	Generación de documentos (memoria del proyecto).	15 horas	mar 06/07/10	jue 15/07/10	34	JP
36	Cierre del proyecto.	1 hora	jue 15/07/10	jue 15/07/10	35	DP;JP
37	Defensa del proyecto.	5 horas	vie 16/07/10	mar 20/07/10	36	JP

Figura 2.1: Planificación de los Recursos Humanos.

Se acompaña la planificación con el diagrama de gantt Figura 2.2 que muestra de manera gráfica el tiempo de dedicación previsto para las diferentes tareas.

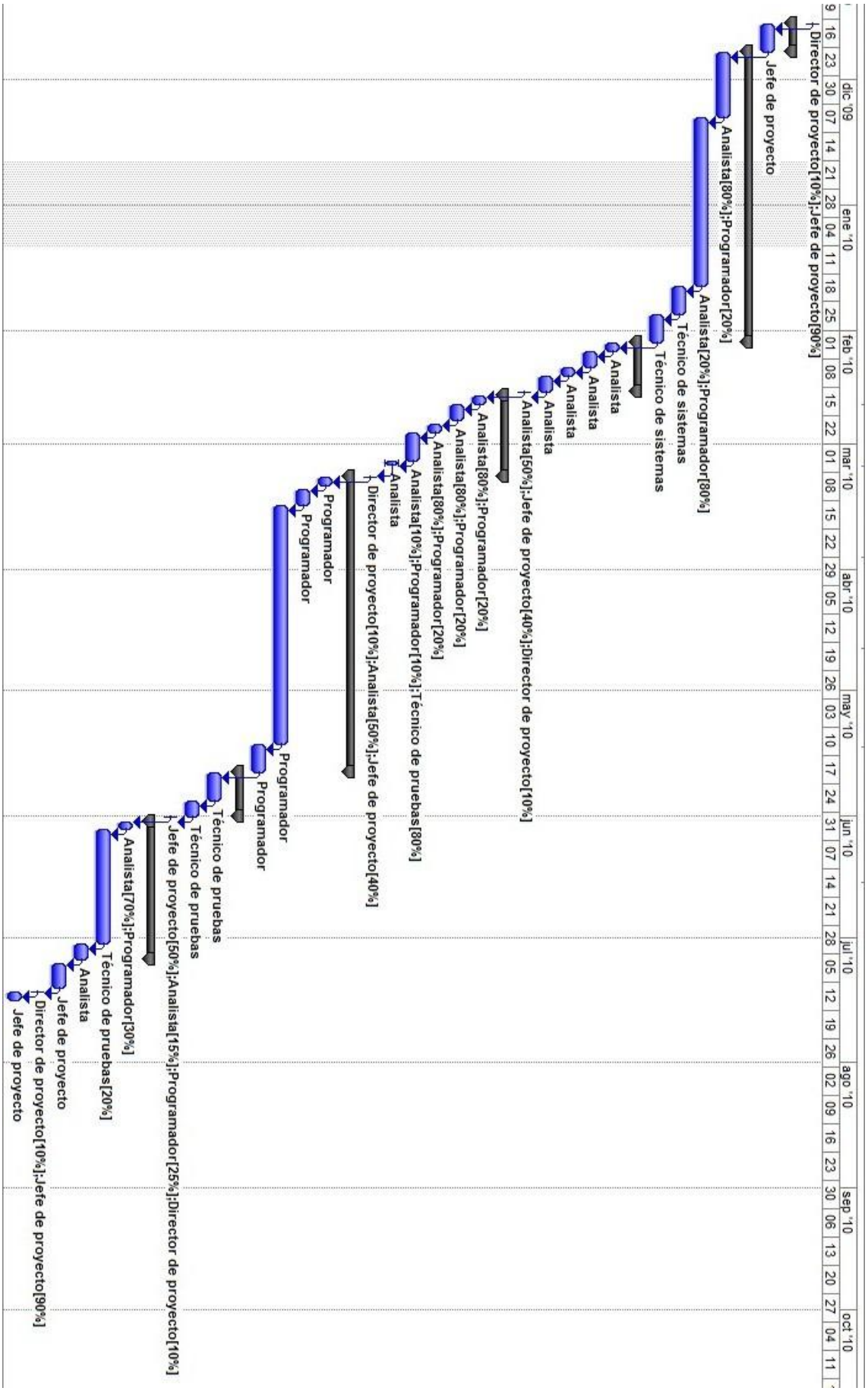


Figura 2.1: Diagrama de gantt.

2.6. Conclusiones

Además de obtener conocimientos sobre una tecnología de programación (Struts) que utiliza una metodología de trabajo (MVC) muy utilizada actualmente, se quiere desarrollar una herramienta que aunque ahora tenga unos costes, en un futuro permitirá realizar cada vez más tipos de webs invirtiendo cada vez menos tiempo en su programación. Otro aspecto a tener en cuenta es que si se consigue realizar una herramienta lo suficientemente potente esta podría ser comercializada u ofrecida en internet como código libre con el objetivo de crear una comunidad que trabaje sobre ella pudiendo obtener un beneficio directo o indirecto.

Analizando los beneficios y los costes se considera que se trata de un proyecto viable.

3. Fundamentos teóricos

En este capítulo se presenta una descripción tanto del framework Struts utilizado para desarrollar el proyecto como de la arquitectura MVC que utiliza este framework. No se pretende redactar una guía práctica que sirva como referencia a la hora de programar sobre la herramienta, sino más bien mostrar: sus conceptos teóricos, componentes que lo integran, funcionamiento general y ventajas que aporta.

3.1. Definiciones previas

3.1.1. Lenguajes de programación web

PHP *(Hypertext Pre-Processor)*: cuando el cliente hace una petición al servidor para que le envíe una página web, el servidor ejecuta el intérprete de PHP que generará el contenido de manera dinámica y se lo envía al cliente. Está considerado como un software de licencia libre y se calcula que se utiliza en más de 20 millones de webs.

Java: es un lenguaje de programación creado por Sun Microsystems, Inc. que permite crear programas que funcionan en cualquier tipo de ordenador y sistema operativo. Las posibilidades que ofrece Java para un entorno web son muy elevadas, ya que el entorno está dividido en dos capas: la programación JSP (*Java Server Pages*) muy similar a la PHP y las Clases Java que ofrecen servicios a las páginas JSP, la ventaja está en estas clases que son programas independientes que se ejecutan en el servidor pudiendo realizar casi cualquier cosa.

HTML *(HyperText Markup Language):* es el lenguaje predominante para la elaboración de páginas web. Su codificación se escribe en forma de etiquetas conocidas como tags, rodeadas por corchetes angulares (<,>) que los navegadores webs interpretan. Este lenguaje es usado para describir la estructura y el contenido en forma de texto, y usando los tags complementar el texto con una serie de objetos como por ejemplo imágenes o tablas que definen gráficamente la apariencia de la web.

Si en el servidor donde está alojada la web permite ejecutar lenguajes de programación interpretados como el PHP o el Java estos lenguajes se incorporan dentro del código HTML una de las diferencias de este lenguaje de cualquiera que sea interpretado es que se ejecuta directamente en el ordenador del usuario, ejecutándose primero el interpretado en el servidor y posteriormente ejecutándose el HTML en el ordenador del usuarios incluyendo el resultado obtenido del servidor.

Javascript: El lenguaje HTML es un lenguaje estático que no ofrece casi ningún tipo de interactividad con el usuario, si se quiere conseguir un poco más de dinamismo en la página web, también se puede incluir en el código HTML. Este lenguaje también se ejecuta en el ordenador del usuario no siendo útil para interacciones con otros entornos externos como por ejemplo una base de datos. Actualmente se utiliza bastante para conseguir efectos visuales.

Css *(Cascade Style Sheet):* lenguaje utilizado por el HTML que describe la presentación de las páginas webs definiéndolo en hojas de estilo separadas del código HTML, es decir, describe cómo se va a mostrar un documento en pantalla, por impresora, y otros dispositivos. Definiendo propiedades tales como los colores, fondos, márgenes, bordes, tipos de letra…

Jquery: biblioteca o framework programada en JavaScript muy popular, que permite simplificar la manera de interactuar con los documentos HTML, manejar eventos, desarrollar animaciones y agregar interacción con las páginas web creando mucho dinamismo mediante efectos visuales. Es un software libre y de código abierto, ofrece una serie de funcionalidades basadas en JavaScript que de otra manera requerirían de mucho más código, es decir, con las funciones propias de esta biblioteca se logran grandes resultados en menos tiempo y espacio.

Mysql: sistema de gestión de bases de datos, su diseño multi-hilo le permite soportar una gran carga de forma muy eficiente. Aunque es software libre, MySQL AB distribuye una versión comercial, que no se diferencia de la versión libre más que en el soporte técnico que se ofrece.

Este sistema también posee un lenguaje de programación exclusivo para la modificación del contenido de las bases de datos.

Este gestor de bases de datos es, probablemente, el gestor más usado en el mundo del software libre, debido a su gran rapidez y facilidad de uso. Esta gran aceptación es debida, en parte, a que existen infinidad de librerías y otras herramientas que permiten su uso a través de gran cantidad de lenguajes de programación, además de su fácil instalación y configuración.

Tomcat: servidor web con soporte de servlets, JSPs y otras aplicaciones java, es el programa donde se incluye la aplicación desarrollada y cuando un usuario acceda con un navegador web a la dirección que apunta a la aplicación el servidor tomcat se encargará de ejecutarla y devolver una respuesta al usuario.

Netbeans: entorno de desarrollo libre y gratuito sin restricciones de uso, hecho principalmente para la edición de programación del lenguaje Java. Ofrece una serie de funcionalidades que ayuda a la hora de estructurar y programar la aplicación ahorrando trabajo en la creación de las clases e indicando errores de código. Otra de las grandes ventajas es que ofrece un entorno de prueba donde ejecutar la aplicación que se está desarrollando.

Navicat: entorno de edición y gestión de bases de datos que ayuda la manipulación de estas conectándose con el servidor de mySQL.

Firebug: extensión del navegador de webs Firefox creada y diseñada especialmente para desarrolladores y programadores web. Es un paquete de utilidades con el que se puede analizar (revisar velocidad de carga, la estructura de las páginas webs), editar, monitorizar y depurar el código fuente, tanto HTML, CSS como JavaScript de manera instantánea. Permitiendo a los desarrolladores mejorar la eficiencia de su código. Tanto el navegador Firefox como su extención Firebug están cosiderados de código abierto, libre y de distribución gratuita.

Programación orientada a objetos: No es un lenguaje específico si no una metodología usada por muchos lenguajes que consiste en un paradigma de programación que usa objetos y las interacciones de estos para diseñar aplicaciones informáticas. Un objeto es una entidad que se puede utilizar desde cualquier parte de la aplicación y que ofrece una serie de valores y funcionalidades. A la hora de programar se definen las clases indicando sus propiedades (variables) y sus métodos (funciones) y en la ejecución del código cuando se instancia (ejecuta) una clase se crea en memoria un objeto representativo de la clase. El conjunto de objetos interactuando entre ellos y dando un servicio al usuario conforman la totalidad de una aplicación.

Una de las ventajas de este tipo de programación es que se obtiene una mejor estructura de la programación organizando y reaprovechando el código mediante clases que ofrecen una serie de características. Por ejemplo si la clase B es muy parecida a la clase A pero con unas pocas diferencias añadidas, se puede hacer que la clase B herede de la clase A, convirtiéndose en una subclase y el único código que se escribirá será el que contenga los añadidos que no tenía la clase A reaprovechando el resto, también se puede sobrescribir código de la clase A que no interese modificándolo en la clase B a nuestro gusto.

Dos tipos de clases/objetos muy utilizadas en la programación web son los Servlets y los Javabeans:

- **Los servles** están ejecutados en el servidor a la espera de alguna petición de acceso a una página web, en el momento que se efectúa dicha petición, por ejemplo un usuario que accede, el servlet lo detecta y ejecuta el procedimiento adecuado para satisfacer la petición.
- **Los javabeans** son utilizados básicamente para transportar información protegida de un lado a otro de la aplicación, solamente están compuestos por una serie de propiedades que guardan la información a los que no se puede acceder directamente y unos métodos que son los encargados de acceder a las propiedades. Con el hecho de para acceder a la información se tenga que pasar por los métodos se consigue una manipulación más segura de la información capando los accesos peligrosos. A esta metodología se le llama encapsulamiento.

3.2. La arquitectura Modelo Vista Controlador

Una aplicación web que no sigue ningún patrón de diseño o modelo de trabajo suele estar organizada siguiendo una arquitectura de tres capas, donde la capa cliente, implementada mediante páginas web, tiene como misión la captura de datos de usuario y su envío a la capa intermedia, así como la presentación de resultados procedentes de ésta. Es la capa intermedia la que constituye el verdadero núcleo de la aplicación, encargándose del procesamiento de los datos de usuario y de la generación de envío de las respuestas a la capa cliente. Durante este proceso, la capa intermedia deberá interaccionar con la capa de datos para el almacenamiento y recuperación de información manejada por la aplicación. Esta organización se representa en la figura 3.1.

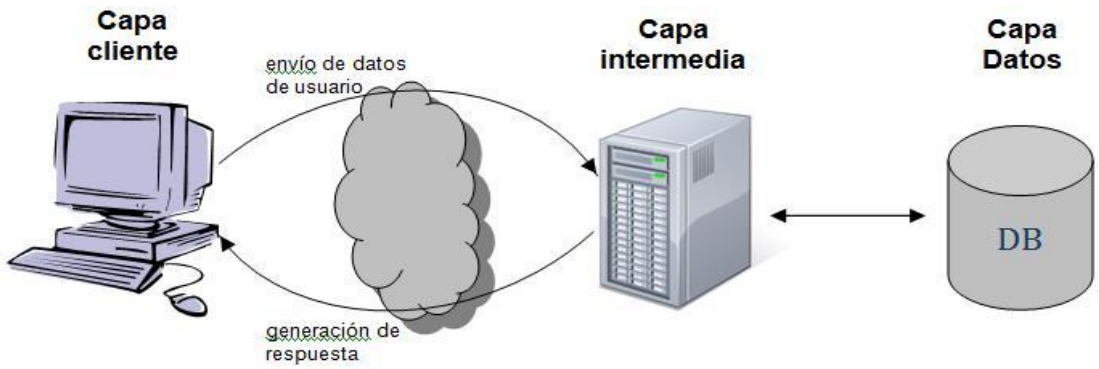


Figura 3.1: Esquema del funcionamiento de las capas de una aplicación web.

Son muchas las tecnologías y lenguajes que los programadores tienen a su disposición para acometer el desarrollo de la capa intermedia de una aplicación (Java, PHP, etc.). No obstante, de cara a afrontar con éxito su implementación, se hace necesario establecer un modelo o esquema que permita estructurar esta capa en una serie de bloques o componentes, de modo que cada uno de estos bloques tenga unas funciones bien definidas dentro de la aplicación y pueda desarrollarse de manera independiente al resto.

Uno de estos esquemas más utilizados por los desarrolladores es la arquitectura Modelo Vista Controlador también conocida como MVC, la cual proporciona una clara separación entre las distintas responsabilidades de la aplicación.

3.2.1. El patrón MVC

Este patrón de diseño especifica cómo debe ser estructurada una aplicación, las capas que van a componer la misma y la funcionalidad de cada una. Indicando que la capa intermedia de una aplicación Web puede ser dividida en tres grandes bloques funcionales como se muestra en la figura 3.2: Controlador, Vista y Modelo.

Controlador

Todas las peticiones a la capa intermedia que se realicen desde el cliente son dirigidas al controlador, cuya misión es determinar las acciones a realizar para cada una de estas peticiones e invocar al resto de los componentes de la aplicación (Modelo y Vista) para que realicen las acciones requeridas en cada caso, encargándose también de la coordinación de todo el proceso.

Por ejemplo, en el caso de que una petición requiera enviar como respuesta al cliente determinada información existente en una base de datos, el Controlador solicitará los datos necesarios al modelo y, una vez recibidos, se los proporcionará a la Vista para que ésta les aplique el formato de presentación correspondiente y envíe la respuesta al cliente.

La centralización del flujo de peticiones en el Controlador proporciona varias ventas al programador, entre ellas:

- Hace que el desarrollo sea más sencillo y limpio.
- Facilita el posterior mantenimiento de la aplicación haciéndola más escalable.
- Facilita la detección de errores en el código.

Vista

Esta capa es la encargada de la generación del formato visual que se enviarán al cliente con los resultados que le envía la capa Controlador obtenidos de la capa Modelo, dicho formato se mostrará en código HTML generado de forma dinámica por una página JSP.

Modelo

En la arquitectura MVC la lógica de negocio de la aplicación, incluyendo el acceso a los datos y su manipulación, está encapsulada dentro de la capa Modelo. Que está formado por una serie de componentes de negocio independientes del Controlador y la Vista, permitiendo así su reutilización y el desacoplamiento entre las capas.

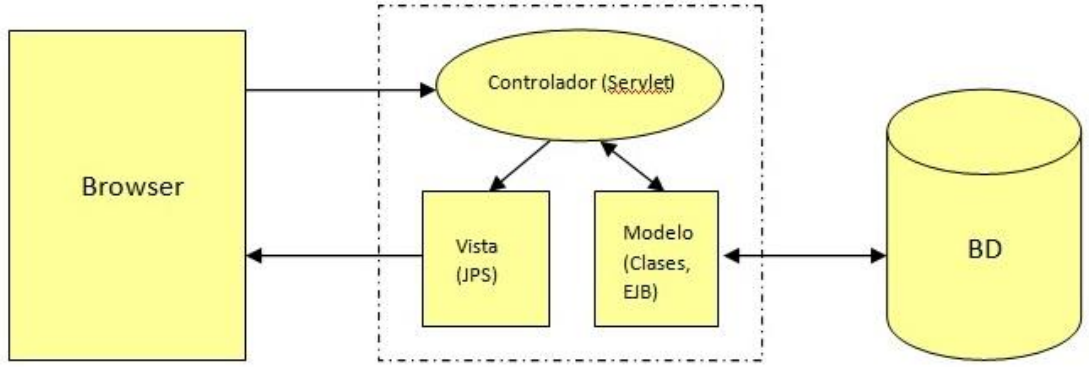


Figura 3.2: Esquema del funcionamiento de una aplicación MVC.

3.2.2. Funcionamiento de una aplicación MVC

Una vez analizados los distintos bloques MVC resulta sencillo comprender el funcionamiento de este tipo de aplicaciones. Para ello, se analizará los procesos que tienen lugar en la capa intermedia desde que llega la petición procedente de la capa cliente hasta que se genera las respuesta:

Captura de la petición en el Controlador: A partir de la URL que se recibe del navegador web utilizado para acceder a la aplicación se determina el tipo de operación que el cliente quiere llevar a cabo. Normalmente, esto se hace analizando el valor de la URL o de algún parámetro que se envía anexado a ella.

Procesamiento de la petición: Una vez que el Controlador determina la operación a realizar, procede a ejecutar las acciones pertinentes, invocando para ello a los diferentes métodos expuestos por el Modelo. Dependiendo de las acciones a realizar, el Modelo necesitará manejar los datos enviados por el cliente en la petición, datos que le serán proporcionados por el Controlador. De la misma manera, los resultados generados por el Modelo serán entregados directamente al Controlador.

Generación de respuesta: Los resultados devueltos por el Modelo al Controlador son depositados en una variable y enviados a la página JSP que debe encargarse de generar la vista correspondiente, esta página accederá a la variable y utilizará los resultados para generar dinámicamente la respuesta HTML que será enviada al cliente.

3.3. El Framework Struts

Struts es un framework o marco de trabajo desarrollado por el grupo Apache, que proporciona un conjunto de utilidades cuyo objetivo es facilitar y optimizar los desarrollos de aplicaciones Web con tecnología Java, siguiendo el patrón MVC.

3.3.1. Fundamentos de Struts

El empleo de Struts en los desarrollos ofrece numerosos beneficios al programador, entre los que se podría destacar:

Control declarativo de peticiones: El programador no tiene que preocuparse de controlar desde código las distintas acciones a realizar en función del tipo de petición que llega al Controlador. Este proceso es implementado por defecto por un de los componentes que proporciona Struts, encargándose únicamente el programador de definir en un archivo de configuración las relaciones entre tipos de petición y acciones a ejecutar.

Utilización de direcciones virtuales: A fin de evitar la inclusión directa dentro del código de la aplicación utilizando las URL que llaman a los recursos, se proporciona unos tipos de objetos que permiten referirse a estos recursos mediante direcciones virtuales asociadas a los mismos. La asociación entre una dirección virtual y su correspondiente URL o dirección real se realiza en un archivo de configuración, de este modo cualquier cambio en la localización de un recurso no afectará al código.

Manipulación de datos con JavaBeans: Los JavaBeans constituyen una pieza importante dentro de la arquitectura MVC al facilitar el transporte de datos entre las capas de la aplicación. Struts proporciona un sólido soporte para la manipulación de datos mediante JavaBeans, encargándose automáticamente el framework de la instanciación de los mismos, su rellenado con los datos procedentes del cliente y la recuperación y almacenamiento de los objetos en las variables de ámbito; todo esto sin necesidad de que el programador tenga que escribir una sola línea de código.

Juego de librerías de acciones JSP: A fin de poder reducir al mínimo la cantidad de instrucciones de código Java dentro de una página JSP, Struts proporciona un amplio conjunto de librerías de acciones predefinidas con las que se pueden realizar la mayor parte de las operaciones que tienen lugar dentro de una página JSP.

La aportación de Struts al desarrollo de aplicaciones MVC se dirige básicamente a la construcción del Controlador y la Vista de una aplicación, dejando total libertad al programador en la implementación del Modelo y pueda así elegir la tecnología que crea más conveniente en cada caso.

Esto no significa que los beneficios de utilizar esta herramienta no se vean relejados también en esta capa, ya que la manera en la que se permite implementar el Controlador ofrece un fuerte desacoplamiento entre éste y el Modelo, al tiempo que facilita una buena cohesión entre ambos.

3.3.2. Componentes de Struts

El marco de trabajo está constituido por los siguientes elementos o componentes: archivos de configuración, el API de Struts y las librerías de acciones JSP.

Archivos de configuración

Una aplicación Struts requiere de unos archivos de configuración adicionales que definen su comportamiento. Entre estos archivos se encuentran:

struts-config.xml: Es el más importante y entre otras cosas, sirve para registrar y configurar los distintos objetos Struts que van a ser utilizados por la aplicación.

applicationResource.properties: Se trata de un archivo de texto plano para el almacenamiento de cadenas de texto en forma de asignación de valores elemento=valor, siendo elemento una clave o identificador asociado y valor el texto que hará referencia a dicho elemento. Cada vez que se requiera hacer uso de alguna de estas cadenas desde la aplicación, se hará utilizando la clave asociada a la misma. De esta manera si el texto tiene que ser modificado el cambio se realizará en el archivo de texto, no en el código. Entre las utilidades de este archivo está la internacionalización de aplicaciones, pudiéndose definir tantos archivos de este tipo como idiomas se quiera mostrar en la aplicación, o el almacenamiento de mensajes de error personalizados.

Este archivo deberá estar definido en el archivo de configuración struts-config.xml para que Struts pueda localizarlo.

validator-rules.xml: Contiene las reglas de los validadores utilizados para la validación automática de los datos de usuario.

validation.xml: Archivo utilizado para la asignación de validadores a los campos de los formularios.

tiles-defs.xml: Los tiles o plantillas representan una de las características más interesantes de Struts de cara a optimizar la creación de vistas. A fin de reutilizar los distintos modelos de plantillas dentro de una aplicación, éstos deberán ser definidos dentro de este archivo.

El API de Struts

Un API (*Application Programming Interface*) es un conjunto de clases de apoyo que el framework proporciona para estructurar las aplicaciones y simplificar su desarrollo. En Struts la mayor parte de estas clases se utilizan en la creación del Controlador y de los JavaBeans para el tratamiento de datos.

Las principales clases que forman este API son:

ActionServlet: Un objeto instanciado de una clase ActionServlet constituye el punto de entrada a la aplicación, recibiendo todas las peticiones HTTP que llegan de la capa cliente. Se trata básicamente de un servlet HTTP cuya clase hereda a HttpServlet. Dicho objeto cada vez que recibe una petición desde el cliente y a fin de determinar la operación a realizar, extrae la última parte de la URL y la contrasta con la información contenida en el archivo de configuración **struts-config.xml**, a partir de la cual, el objeto lleva a cabo la instanciación del JavaBeans (ActionForm) asociado a la acción, lo rellena con los datos procedentes del formulario cliente y deposita la instancia en el contexto correspondiente, pasando a continuación el control de la petición al objeto Action encargado de procesarla.

Action: Estos tipos de objetos son los responsables de procesar los distintos tipos de peticiones que llegan a la aplicación. El principal método con el que cuenta esta clase es **execute()**, que será invocado por ActionServlet al transferir la petición al objeto. Así pues, por cada tipo de petición que se vaya a controlar, el programador deberá definir una subclase de Action y sobrescribir el método **execute()**, incluyendo en él las instrucciones requeridas para el tratamiento de la petición, tales como llamadas a los métodos de la lógica de negocio implementada en el modelo o la transferencia de resultados a las vistas para su presentación.

Además de ésta clase se incluyen otras sub clases de Action cuya utilización puede ser más adecuada en determinadas aplicaciones a la hora de gestionar ciertos tipos de peticiones, de modo que en vez de crear una subclase de Action para su tratamiento se crearía un subtipo de una de estas otras clases:

- **DispatchAction:** Permite definir en la misma clase diferentes métodos para tratar un grupo de peticiones similares, evitando tener que crear una clase distinta para la gestión de cada acción. Utilizándose el valor de un parámetro enviado con cada petición para determinar el método que se tiene que ejecutar.
- **LookupDispatchAction:** Subclase de la clase DispatchAction que, al igual que ésta, tiene como misión gestionar varias peticiones en una misma clase mediante la definición de un método personalizado para cada acción. A diferencia de la clase

DispatchAction donde el valor del parámetro contiene directamente el nombre del método asociado a la acción, LookupDispatchAction utiliza el valor de este parámetro para localizar en el archivo de recursos ApplicationResource.proerties una clave asociada al mismo entre todas las parejas clave=valor amacenadas.

- **MappingDispatchAction:** Al igual que las anteriores, permite definir en la misma clase un grupo de métodos para la gestión de peticiones diferentes. En teste caso no se utiliza ningún parámetro que permita determinar el método a ejecutar, sino que cada petición utilizará un path diferente para acceder al mismo objeto, lo que obligará a definir el fichero de configuración struts-config.xml tantos elementos <action> asociados a la misma subclase de MappingDispatchAction como peticiones distintas deba gestionar el objeto.

ActionForm: Los objetos de este tipo son un tipo especial de JavaBean que facilitan el transporte de datos entre las capas de la aplicación. Son utilizados por ActionServlet para capturar los datos preocedentes de un formulario HTML y enviárselos al objeto Action correspondiente, todo ello sin recurrir a los incómodos request.getParameter(). Para ello el programador deberá extender esta clase y proporcionar los datos miembros necesarios para el almacenamiento de los datos, así como los correspondientes métodos set/get que den acceso a los mismos.

ActionMapping: Un objeto de tipo ActionMapping representa una asociación entre una petición y el objeto Action que la tiene que procesar. Contiene información sobre el path o tipo de URL que provoca la ejecución de la acción, así como de las posibles vistas que se pueden presentar al cliente tras su procesamiento. Cuando el Controlador ActionServlet invoca al método **execute()** de un objeto Action para el procesamiento de una petición, proporciona como parámetro un objeto ActionMapping con la información asociada a la misma, pudiéndose hacer uso de sus métodos para encaminar al usuario a cualquiea de las vistas asociadas al objeto una vez que la petición ha sido procesada.

ActionForward: Las direcciones virtuales más conocidas en Struts como **forwards**, son manejadas desde código a través de objetos ActionForward, encapsulando los detalles sobre la localización de un recurso, de modo que cada vez que se quiera transferir una petición desde código o re-direccionar al usuario a una determinada página se hará utilizando estos objetos.

Si en la aplicación se utiliza cualquiera de estos objetos definidos por el framework Struts se deberán definir apropiadamente en el archivo de configuración **struts-config.xml**.

Librerías de acciones JSP

Las librerías de tags o acciones JSP constituyen otro de los componentes esenciales se Struts. El amplio conjunto de acciones existentes permite realizar la gran mayoría de las tareas propias de una página JSP sin necesidad de incluir en la misma ni una sola línea de código Java, facilitando así la comprensión y posterior mantenimiento de las vistas.

Struts proporciona las siguientes librerías:

HTML: Incluye acciones para la construcción de formularios HTML, incorporando parte de la funcionalidad encargada de la creación de beans de tipo ActionForm y el rellenado de los mismos con los valores de los controles. Además de este tipo de acciones, la librería proporciona otros tags que facilitan la realización de tareas habituales en una página Web, como el enlace a otros recursos de la aplicación o la visualización de mensajes de error.

Bean: Esta librería incluye acciones para el acceso a propiedades de un bean y parámetros de la petición desde una página JSP. También incluye acciones para la definición de nuevos beans y su almacenamiento en el contexto de la aplicación.

Logic: Los tags incluidos en esta librería realizan las operaciones que normalmente se llevan a cabo utilizando las estructuras lógicas de un lenguaje de programación, como el recorrido de una colección o la evaluación de condiciones.

Tiles: Esta librería incluye acciones para la definición de plantillas y su reutilización den diferentes páginas.

3.3.3. Funcionamiento de una aplicación Struts

Cada vez que desde el navegador cliente llega una petición al contenedor tendrá lugar el proceso que se describe a continuación:

Análisis de la URL de la aplicación: El contenedor Web pasa la petición al objeto ActionServlet, éste, utiliza la última parte de la URL de la petición para determinar la acción a realizar. Es habitual asignar como comodín del objeto ActionServlet el valor *.do, esto significa que cualquier URL procedente del cliente que termine en .do provocará que la petición sea capturada por este servlet.

Determinación de la acción a realizar: Utilizando el dato obtenido en el paso anterior, el objeto ActionServlet realiza una consulta en el archivo **struts-config.xml** para determinar las operaciones a realizar. Para cada tipo de acción el archivo de configuración define la subclase Action que debe ser instanciada, así como el objeto ActionForm asociado a la operación. Tras realizar la consulta en el archivo, ActionServlet lleva a cabo las siguientes acciones:

- Crea u obtiene la instancia del objeto ActionForm y lo rellena con los datos del formulario cliente.
- Crea una instancia del objeto Action correspondiente e invoca a su método **execute()**, pasándole como parámetro una referencia al objeto ActionForm.

Generación de la vista: Como resultado de su ejecución, el método **execute()** devuelve a ActionServlet un objeto ActionForward que identifica al recurso utilizado para la generación de la respuesta. A partir de este objeto, Action Servlet extrae la dirección virtual encapsulada en el mismo y utiliza este valor para obtener del archivo struts-config.xml la dirección real de la página HTML o JSP correspondiente.

3.3.4. Validación de datos de usuario

La utilización de validadores en una aplicación Struts proporciona un mecanismo de validación automática, de tipo declarativo, que evita la codificación de estas tareas desde el código de programación, permitiendo además la validación de los datos tanto en el ordenador cliente como en el servidor.

La utilización de validadores en una página JSP se apoya en el empleo de una serie de componentes que proporcionan toda la funcionalidad necesaria para llevar a cabo la validación automática y que habrá que configurar adecuadamente en cada aplicación, estos componentes son:

- Plug-in validator: es la clase incorporada en Struts encargada de gestionar las validaciones automáticas dentro de una aplicación Web. Para que puedan utilizarse las validaciones automáticas en una determinada aplicación será necesario registrar esta clase en el archivo de configuración struts.config.xml.
- Archivos de configuración: Además del fichero struts-config.xml, las aplicaciones que utilizan validadores necesitan dos archivos de configuración adicionales:
    - validator-rules.xml: a fin de que el programador no tenga que codificar las instrucciones para la validación de los datos, Struts incorpora una serie de clases predefinidas cuyos métodos realizan las tareas habituales de validación de datos de usuario requeridas por la mayoría de las aplicaciones Web. Este fichero contiene la declaración de estas clases de validación así como los métodos utilizados para realizar cada una de las rutinas de validación predefinidas.
    - validation.xml: en este archivo se asignará a cada campo cuyos datos se quiera validar la regla o reglas de validación definidas en el archivo validator-rules.xml. Para cada formulario que se desee validar se definirá un elemento cuyo atributo deberá contener el nombre del objeto ActionForm encargado de capturar el datos de usuario.
- La clase ValidatorForm: Si se quiere utilizar la validación automática de datos de usuario que proporciona Struts, a la hora de definir la clase JavaBean en la que se recogerán los datos procedentes del formulario cliente se debe utilizar como clase base ValidatorForm en vez de ActionForm que es a su vez una subclase de ActionForm.
- Archivo de recursos ApplicationResource.properties: se utiliza en para almacenar cadenas de texto a las que se les asocia una clave, cadenas que pueden tener diferentes usos dentro de la aplicación. En el case concreto de los validadores se hace uso de este archivo de recursos para almacenar los mensajes de error devueltos por cada validador cuando se incumpla el criterio de validación definido para los campos.

3.3.5. Utilización de plantillas

Las plantillas también conocidas en Struts como tiles, tienen como objetivo posibilitar la reutilización de código, XHTML/JSP durante el desarrollo de la Vista en una aplicación Web MVC, simplificando el desarrollo de las páginas y haciendo también más cómodos y menos propensos a errores los posibles cambios que se puedan realizar a posteriori sobre esta capa de la aplicación.

La utilización de plantillas resulta adecuada en las aplicaciones web en las que se tiene un grupo de páginas con estructura similar, en las que el contenido de ciertas partes de las mismas es igual en todas ellas.

Para utilizar plantillas en una aplicación Struts es necesario indicarlo en el fichero de configuración struts.config.xml.

4. Análisis de requerimientos

En este capítulo se detallan los requerimientos de la descripción del proyecto. Posteriormente se realiza el análisis de la aplicación que se va a desarrollar, mediante diagramas de casos de uso y el diagrama de clases.

4.1. Requerimientos funcionales

La aplicación que se desarrolla está dividida en dos partes:

- El CMS que permitirá definir y administrar la web, cuyo acceso estará restringido a los usuarios con permiso para acceder.
- Y la web generada por el CMS que podrá contener secciones pública para todos los tipos de usuarios o secciones que solamente podrán acceder los tipos de usuarios que se han indicado desde el CMS.

La parte del CMS está dividida en cinco secciones: Estructuras, Apartados, Usuarios, Permisos, Apariencias y Formularios.

Estructuras: dentro de esta sección se pretende definir los tipos de contenidos que tendrá la web, definiendo su esqueleto mediante Componentes, por ejemplo, en el caso de una noticia se definirá un título, un resumen, un texto y una foto, indicando el espacio que ocupa cada componente dentro de la pantalla.

Apartados: Aquí es donde se introducirá el contenido que se verá en la web, por ejemplo, será donde se introducirá las diferentes noticias.

Usuarios y Permisos: Sección donde se administrarán los diferentes usuarios que pueden acceder tanto al CMS como a la web que se generará.

Permisos: En esta sección se definirán las restricciones que tendrán cada tipo de usuario.

Apariencia: También se ofrecerá la opción de administrar el diseño general de la web, colores, fondos de pantalla, tipografía de los textos, etc.

Recolección de formularios: Como contenido de la web se ofrecerá la opción de definir formularios que los usuarios podrán usar desde la web. En esta sección es donde se podrá ver los datos que se han enviado a través de los formularios.

En principio se realizará la aplicación pensando en poder crear dos tipos de webs diferentes, la primera será una web pública que tratará sobre la promoción de un pueblo de Burgos y la segunda será un espacio virtual de acceso restringido que servirá de agenda a un alumno universitario.

4.2. Requerimientos no funcionales

En la realización de la aplicación se deberá cumplir una serie de requerimientos no funcionales:

- Realizar la aplicación mediante el framework struts.
- La programación de la aplicación deberá utilizar la estructura de trabajo MVC.
- Deberá ofrecer una seguridad de prevención de errores y acciones incorrectas de la aplicación.
- Deberá controlar todas las entradas de los usuarios permitiendo las que sean legales y evitando las que no.
- La aplicación deberá ofrecer un servicio rápido y poco pesado que consuma la mínima memoria en las máquinas físicas que accedan a ella.
- Realizar una programación modular que en un futuro permita incorporar nuevas funcionalidades que amplíen la capacidad de realizar un número más elevado de tipos de webs.

4.3. Diagrama de casos de uso

Para describir la aplicación que se implementará desde el punto de vista del usuario, a continuación se muestran todas las interacciones posibles del usuario con la aplicación utilizando diagramas de casos de uso.

4.3.1. Identificar tipo de usuario

Al acceder a la aplicación el usuario se deberá identificar mediante un login y una contraseña, tal como se muestra en la figura 4.1. Si realiza esta identificación correctamente mientras navegue por la aplicación se recordará el Tipo de usuario al que pertenece y tendrá unos privilegios y unas limitaciones determinadas.

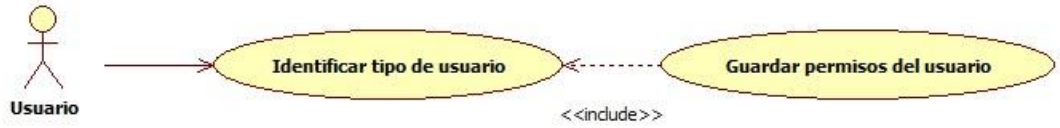


Figura 4.1: Diagrama de secuencia de identificación de tipo de usuario.

4.3.2. Administrar Estructuras

Desde esta sección se tiene la posibilidad de listar, pre-visualizar, crear, modificar y eliminar las Estructuras que componen todos los tipos de contenidos que puede tener la web generada por la aplicación. Cuando se crean o modifican se indican sus propiedades, entre las que se encuentran una serie de componentes que forman el contenido de una Estructura. Los

formatos de los Componentes están definidos por la programación de la aplicación. Dentro de una Estructura se tiene la opción de listar, crear, modificar y borrar sus Componentes. Las estructuras se agrupan dentro de grupos de estructuras teniendo que acceder a un grupo para poder llegar a ellas, estos grupos se pueden listar, crear, modificar, eliminar y acceder a ellos. En la figura 4.2 se muestra el procedimiento de la administración de una estructura.

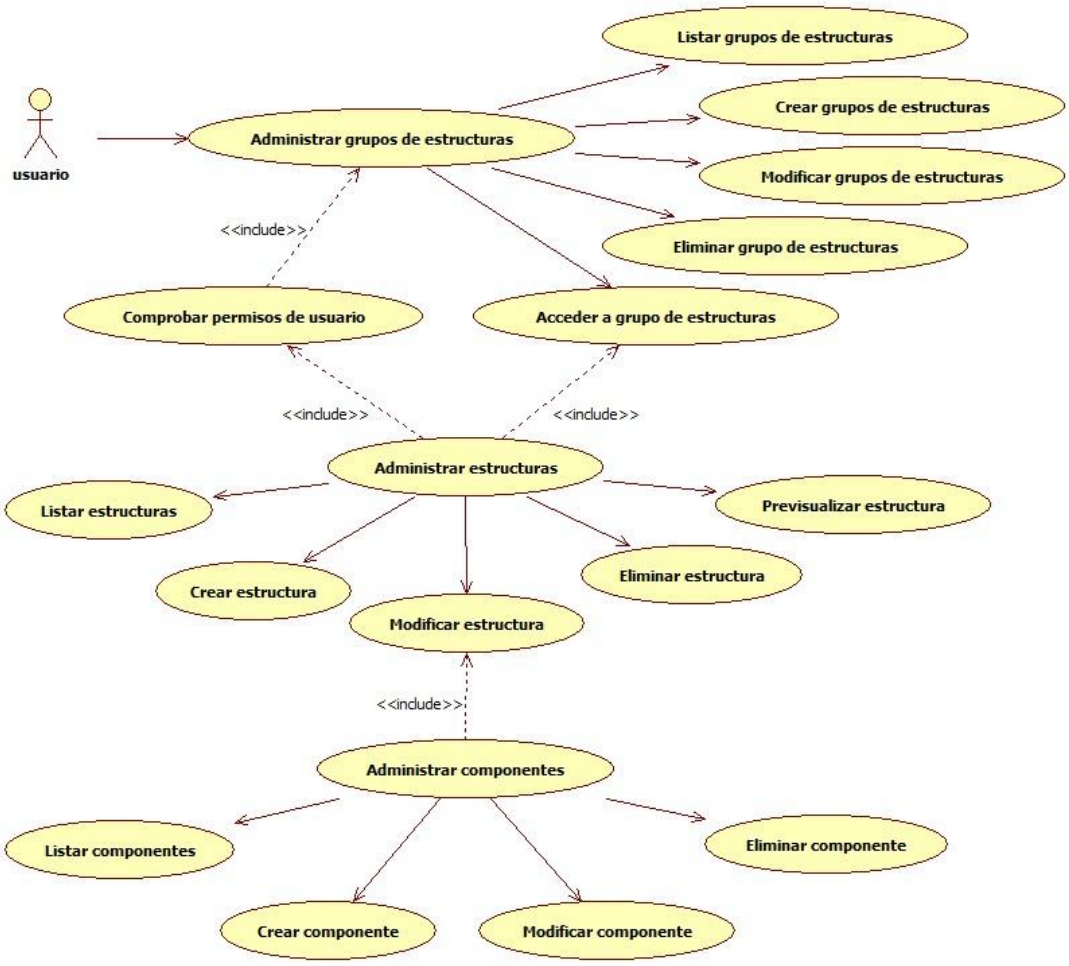


Figura 4.2: Diagrama de secuencia de la administración de una estructura.

4.3.3. Administrar Apartados

Tal y como se muestra en la figura 4.3 los apartados se pueden listar, pre-visualizar, crear, modificar y eliminar. También se puede acceder dentro de un apartado para crear más apartados dentro de él, creando así, un listado de contenidos en formato de ramas de árbol que posteriormente se verá reflejado en la web como secciones que contienen sub-secciones.

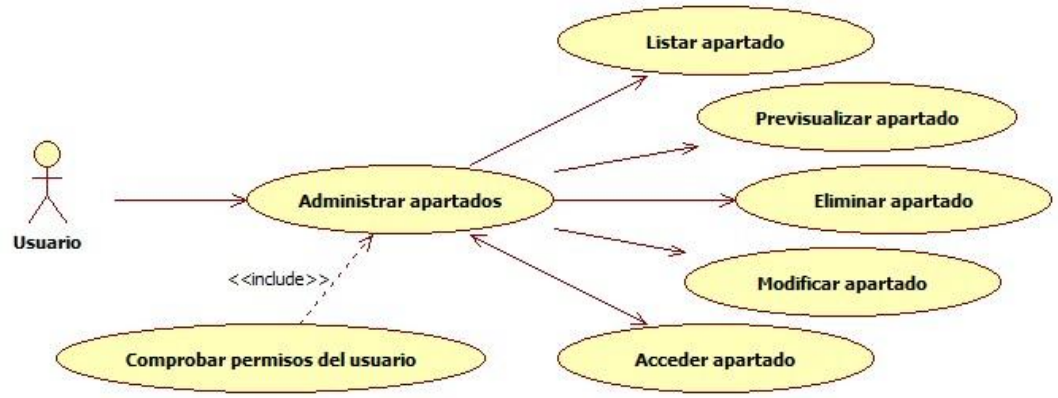


Figura 4.3: Diagrama de secuencia de la administración de un apartado.

4.3.4. Administrar Permisos

Los Permisos son conjunto de restricciones que definen los privilegios de acceso de los diferentes tipos de usuario tanto en la aplicación como en la web definida. Se pueden restringir accesos tanto a secciones de la administración como a tipos de apartados definidos por las estructuras. Los tipos de restricciones pueden ser de lectura, de acceso y de escritura. Los permisos están agrupados en grupos de permisos teniendo que acceder a un grupo para llegar a ellos. Los grupos de permisos dan la opción de listar, crear, modificar, eliminarlos. La secuencia de la administración de permisos se explica gráficamente en la figura 4.4.

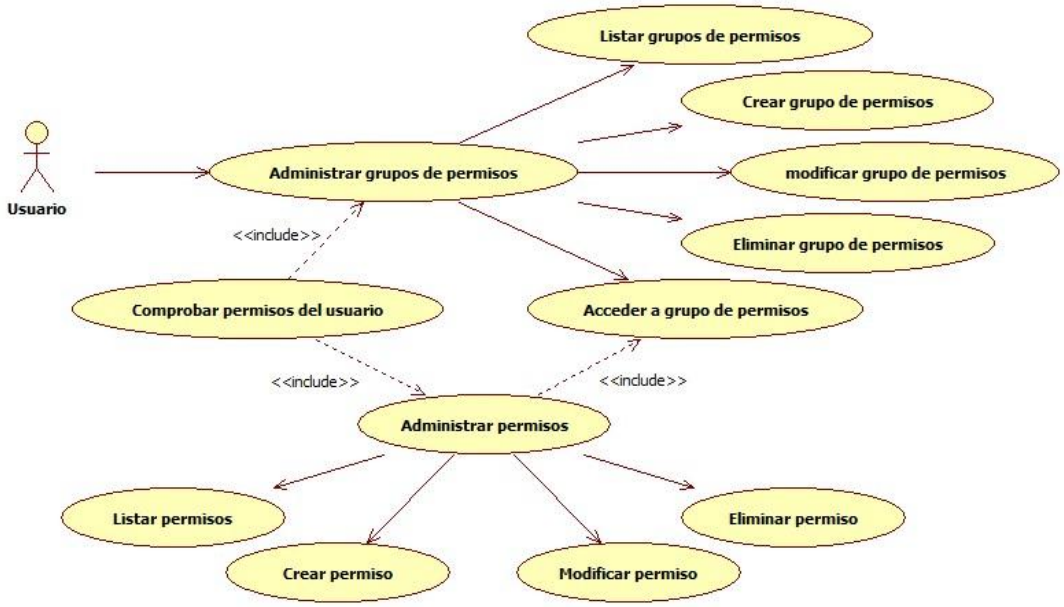


Figura 4.4: Diagrama de secuencia de la administración de un permiso.

4.3.5. Administrar Usuarios

Se ofrece las funcionalidades de listar, crear, modificar y eliminar los usuarios que pueden acceder a la aplicación. Para acceder a los usuarios hay que entrar dentro del grupo de Usuarios que los agrupa. Los usuarios obtienen los privilegios y restricciones del Grupo al que pertenece.

Los Grupos de Usuarios son contenedores que sirven para agrupar una serie de Usuarios relacionadas entre sí. Dan la opción de listar, crear, modificar, eliminarlos y acceder dentro del grupo para poder administrar los Usuarios que contiene. La secuencia de la administración de usuarios se puede ver en la figura 4.5.

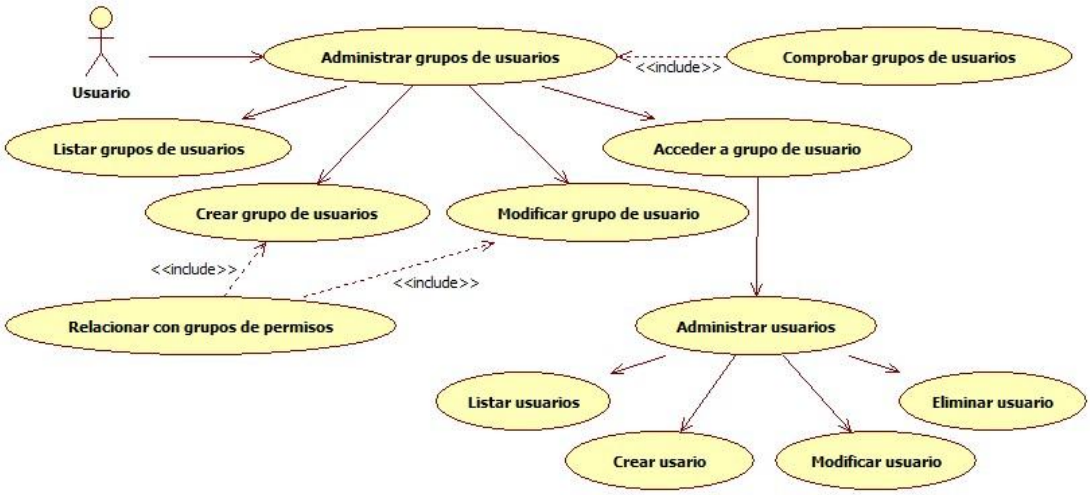


Figura 4.5: Diagrama de secuencia de la administración de un usuario.

4.3.6. Administrar Apariencias

La sección de Apariencias da la opción de listar, crear, modificar y eliminarlas ver figura 4.6.

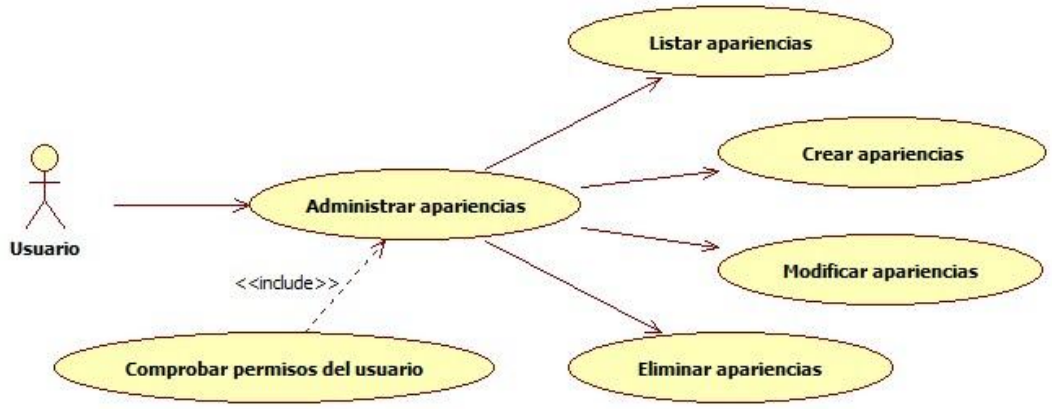


Figura 4.6: Diagrama de secuencia de la administración de una apariencia.

4.3.7. Administrar Formularios y recolecciones de formularios

Los formularios son contenedores que sirven para agrupar una serie de Recolecciones de formularios con los datos que han enviado los usuarios a través de los formularios definidos en la web. La sección de Formularios da la opción de acceder, listar, crear, modificar y eliminarlos.

Las Recolecciones de Formularios son los datos que pertenecen a un envío particular de un usuario que ha utilizado uno de los formularios definidos en la web. La sección de Formularios da la opción de listar, modificar y eliminarlos. Ver figura 4.7.

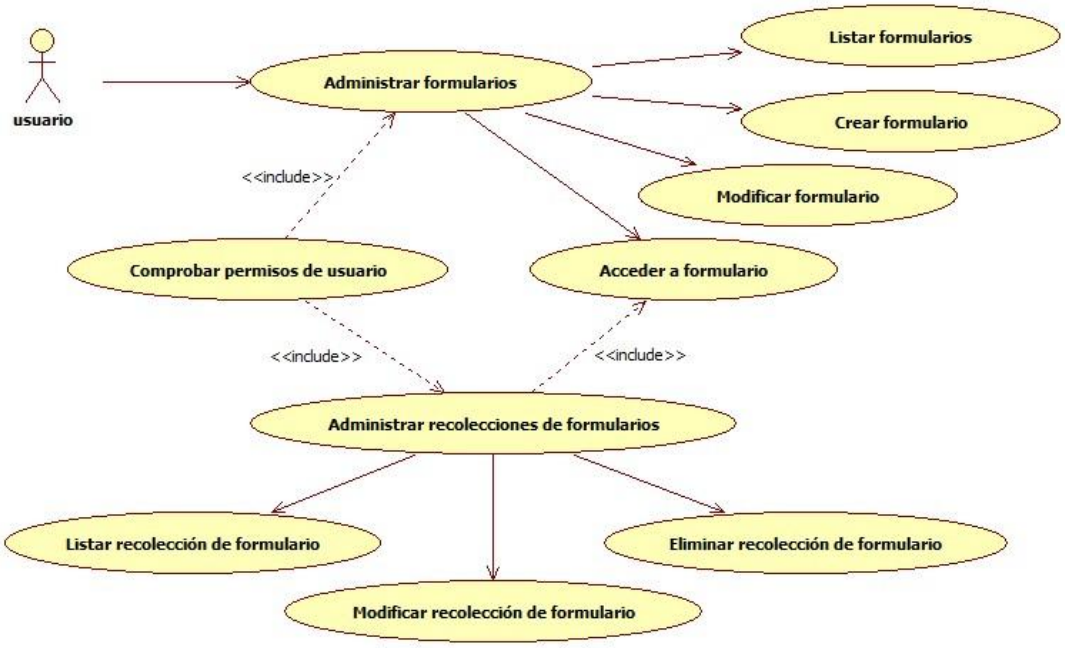


Figura 4.7: Diagrama de secuencia de la administración de un formulario y sus recolecciones de formulario.

4.3.8. Acceder a la web

En la web definida por la aplicación se pueden crear secciones o apartados de acceso restringido a los cuales solamente podrán acceder los usuarios que se les hayan concedido permisos.

Por último se tiene al tipo de usuario anónimo que sería cualquier usuario no incluido en uno de los tipos mencionados y el único privilegio que posee es acceder y leer las zonas totalmente públicas. Ver diagrama 4.8.

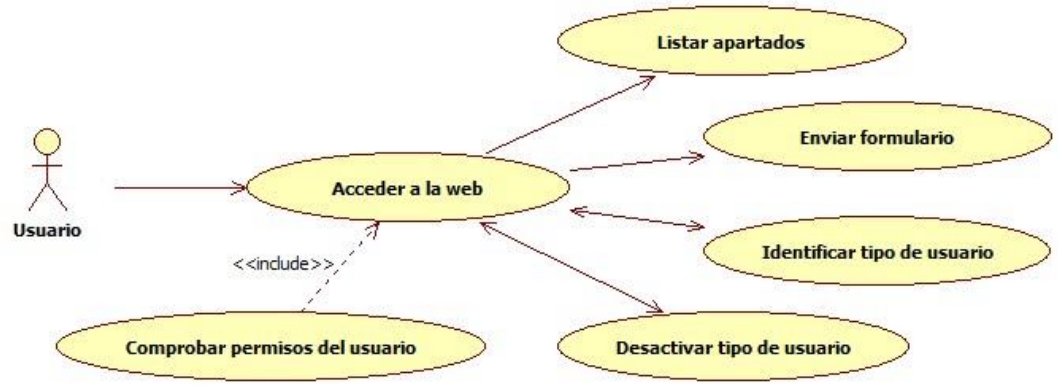


Figura 4.8: Diagrama de secuencia del acceso a la web generada por la aplicación.

4.4. Diagrama de clases

En este apartado se muestran las estructuras de las clases y las relaciones entre ellas necesarias para el desarrollo de la aplicación. Las clases se pueden agrupar en varios grupos dependiendo de su finalidad:

- Las clases que tienen la nomenclatura Action en el nombre son las que al ejecutarse la aplicación estarán esperando una petición del usuario, encargándose de identificarla e interactuar con el resto de las clases para resolver dicha petición.
- Las clases que tienen la nomenclatura Form en el nombre son las que utiliza el framework Struts para recoger los datos que envía el usuario a través de un formulario, evitando al desarrollador tener que programar esta recogida de datos.
- Las clases que tienen la nomenclatura Bean en el nombre se utilizan únicamente para transportar información de una clase a otra.
- Las clases que tienen la nomenclatura Exception en el nombre son las que utiliza el framework para detener la ejecución de una petición y redirigir a otra página web.
- Las clases que tienen la nomenclatura Gestion en el nombre son las que ha programado el desarrollador y pretenden realizar las funcionalidades de la aplicación.

4.4.1. Validación del usuario

Cuando el usuario realice una identificación de login y contraseña a través de un formulario, se utilizará la clase ValidacionAction Figura 4.9, la cual recogerá los datos de la validación a través de la clase UsuariosForm, le pedirá a la clase GestionUsuarios que compruebe si el login y la contraseña son correctos, guardando la correcta validación y los permisos que tiene en la clase UsuariosBean, para que en el resto de las peticiones que efectué el usuario se pueda comprobar que ha realizado la validación y qué tipo de restricciones tiene. Si la validación no ha sido correcta se interrumpirá la ejecución a través de la clase UsuarioIncorrectoException. Para que la clase GestionUsuarios pueda comprobar la validez del login y password utilizará la clase GestionBaseDeDatos que se conectará a la base de datos.

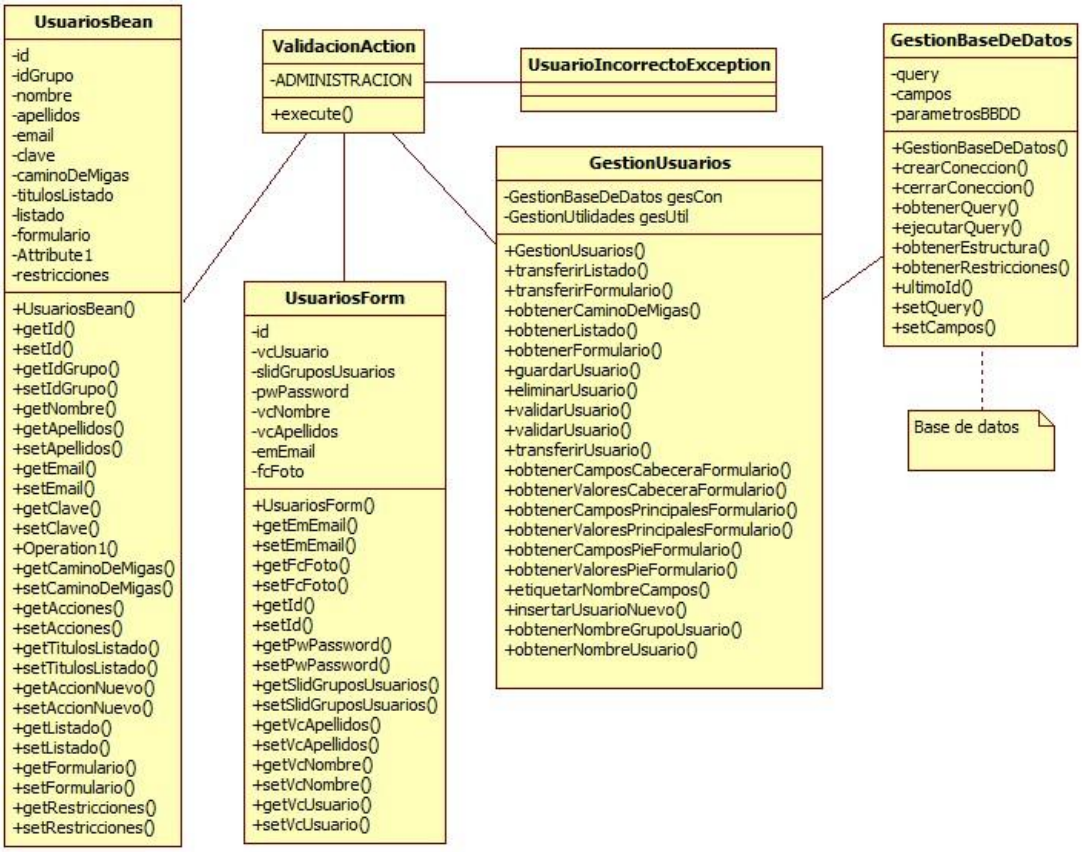


Figura 4.9: Validación del usuario.

4.4.2. Administración de las secciones

Cuando un usuario acceda a una sección de la administración lo primero que verá será un listado de los elementos que la componen. Para generar estos listados se utilizará la clase AdministraciónAction, la cual, primero comprobará en la clase UsuariosBean si se ha realizado la validación de usuario y si se tiene suficientes permisos como para acceder a la sección. Si no puede acceder a la sección se interrumpirá la petición a través de la clase UsuarioIncorrectoException. Si puede acceder, la clase AdministracionAction utilizará la clase Gestion correspondiente a la sección donde se quiere acceder para obtener los datos de los listados, los guardará en la clase Bean correspondiente (en el caso de la figura 4.10 la sección se corresponde a Estructuras) para enviarlos a la capa Vista de la aplicación para que muestre el listado al usuario.

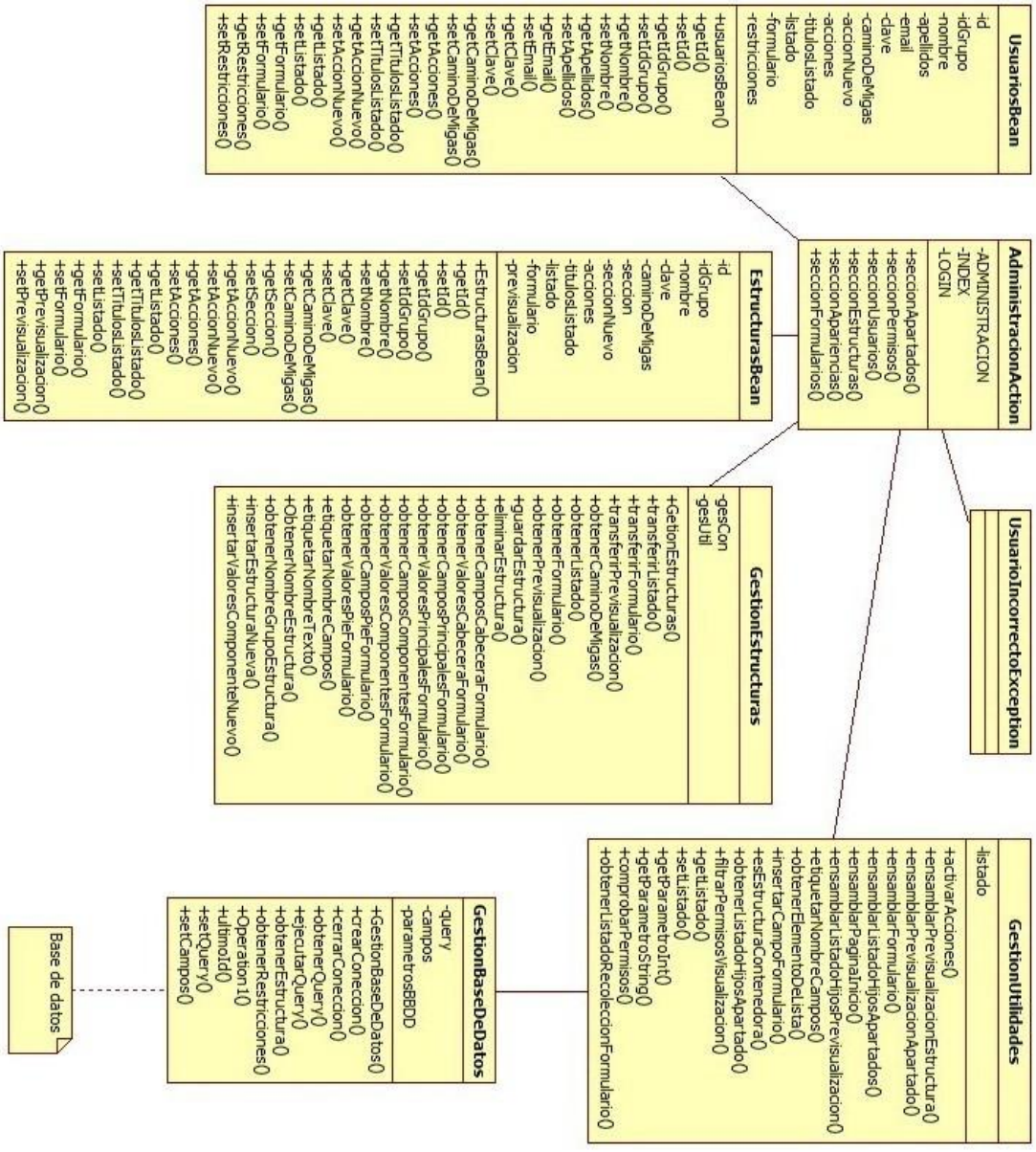


Figura 4.10: Administración de las secciones.

4.4.3. Edición de las secciones

Cuando un usuario acceda a un formulario de edición de una sección se utilizará la clase EdicionAction que gestionará la petición para crear el formulario, la cual, primero comprobará en la clase UsuariosBean si ha realizado la validación de usuario y si tiene suficientes permisos como para acceder al formulario de la sección. Si no puede acceder se interrumpirá la petición a través de la clase UsuarioIncorrectoException. Si puede acceder, la clase EdicionAction utilizará la clase Gestion correspondiente a la sección donde se quiere acceder al formulario, para obtener los datos necesarios, los guardará en la clase Bean correspondiente (en el caso de la figura 4.11 la sección se corresponde a Estructuras) para enviarlos a la capa Vista de la aplicación para que muestre el formulario al usuario.

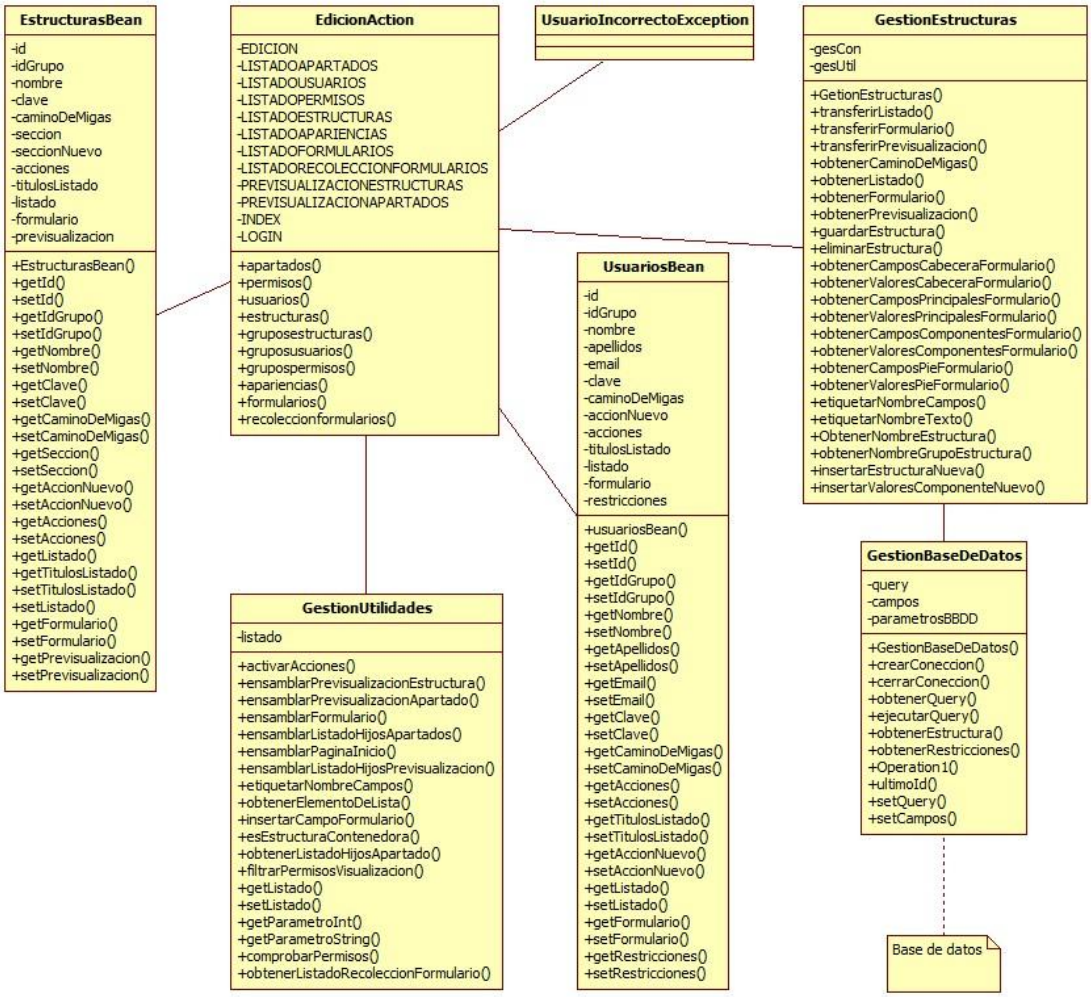


Figura 4.11: Edición de las secciones.

4.4.4. Guardar el elemento de una sección

Cuando un usuario utilice un formulario de edición de una sección para modificar un elemento, se utilizará la clase GuardarAction que gestionará la petición para modificarlo, la cual, primero comprobará en la clase UsuariosBean si ha realizado la validación de usuario y si tiene suficientes permisos como para modificar un elemento de la sección. Si no puede modificar el elemento se interrumpirá la petición a través de la clase UsuarioIncorrectoException. Si puede modificarlo, la clase GuardarAction utilizará la clase Gestion correspondiente a la sección del elemento que se quiere modificar, para que guarde los datos en la base de datos (en el caso de la figura 4.12 la sección del elemento se corresponde a Estructuras).

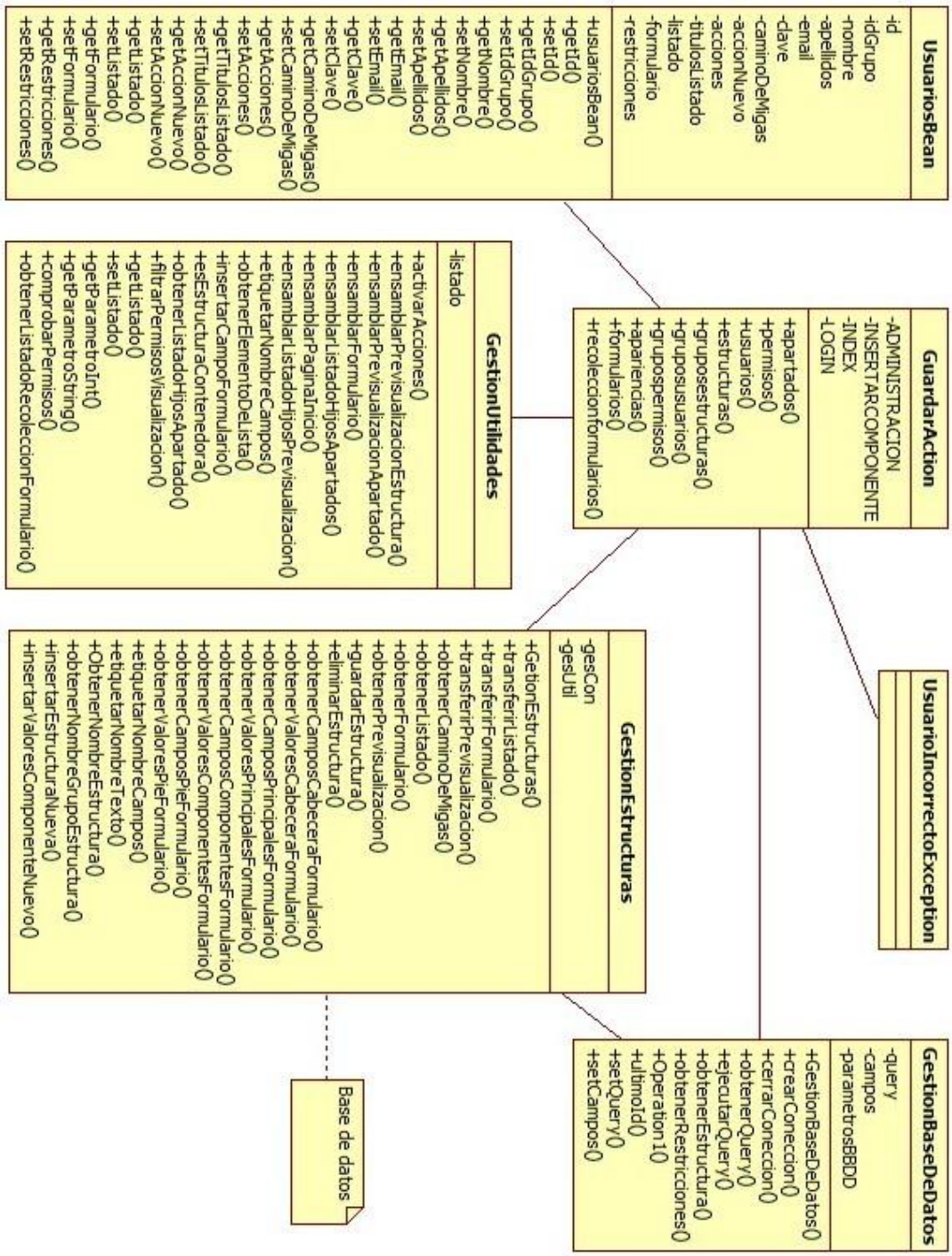


Figura 4.12: Guardar el elemento de una sección.

4.4.5. Acceso a un apartado de la web

Cuando un usuario acceda a un apartado de la web generada por el CMS se utilizará la clase WebAction que gestionará la petición para mostrar el contenido Figura 4.13, la cual, primero comprobará en la clase UsuariosBean si ha realizado la validación de usuario y si tiene suficientes permisos como para acceder al apartado de la web se que quiere ver. Si no ha realizado la validación se utilizará la clase GestionUsuarios para otorgarle los permisos de un usuario anónimo el cual no podrá acceder a ningún apartado con restricciones. Si puede acceder al apartado, la clase WebAction utilizará la clase GestionApartados para obtener los contenidos del apartado y la clase GestionApariencias para obtener la apariencia gráfica de la web, guardando los datos en las clases correspondientes ApartadosBean y AparienciasBen para poder enviarlos a la Capa Vista de la aplicación y que finalmente le muestre al usuario el apartado de la web donde ha accedido.

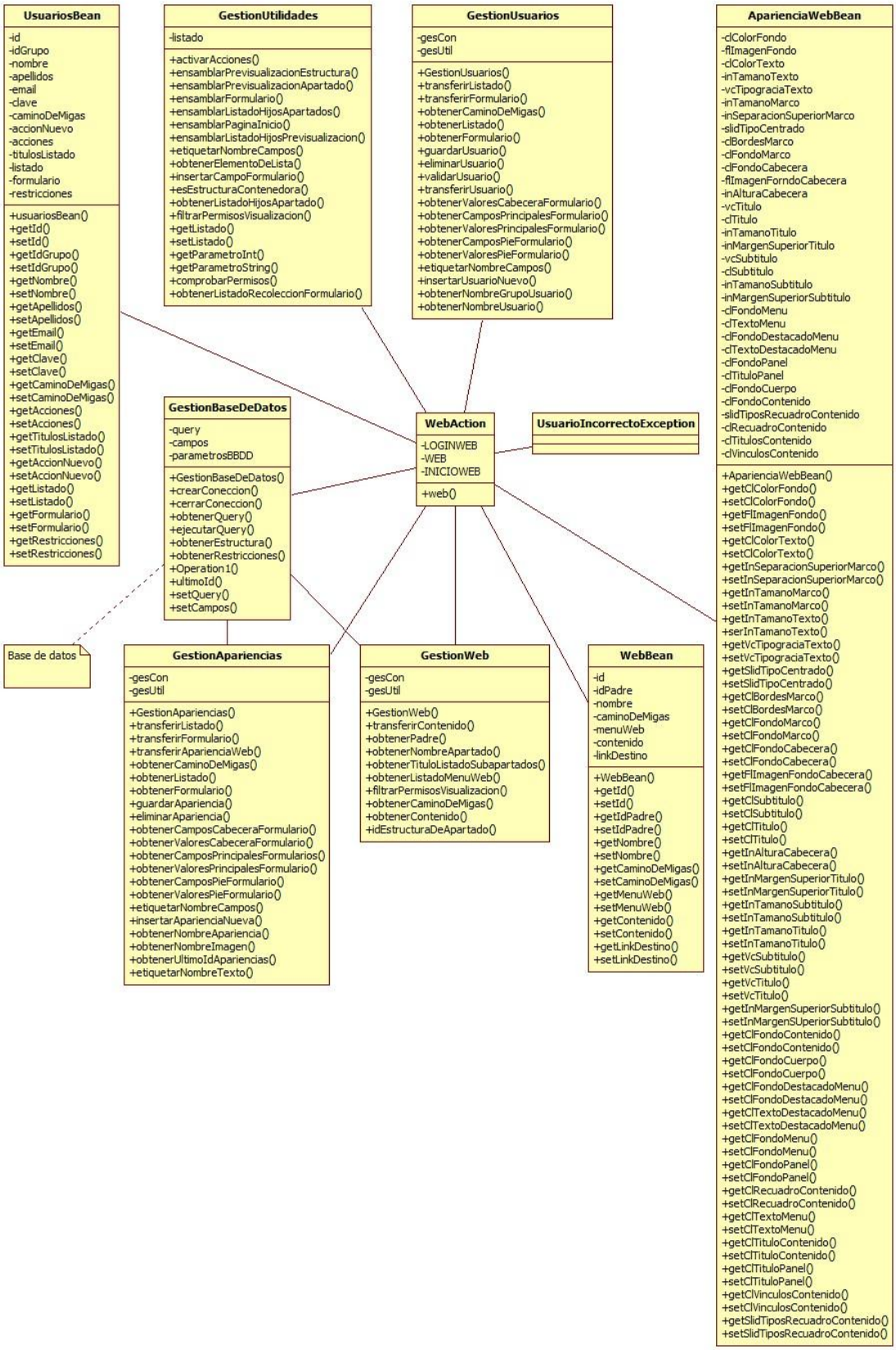


Figura 4.13: Acceso a un apartado de la web.

4.4.6. Envío de datos a través de un formulario de la web generada

Cuando un usuario envía información a través de un formulario de la web generada por el CMS se utilizará la clase FormularioAction que gestionará la petición para guardar la información enviada Figura 4.14, la cual, primero comprobará en la clase UsuariosBean si ha realizado la validación de usuario y si tiene suficientes permisos como para utilizar el formulario de la web. Si no ha realizado la validación se utilizará la clase GestionUsuario para otorgarle los permisos de un usuario. Si puede utilizar el formulario, la clase FormularioAction utilizará la clase GestionFormularios para guardar la información que se ha recolectado en la clase FormulariosForm.

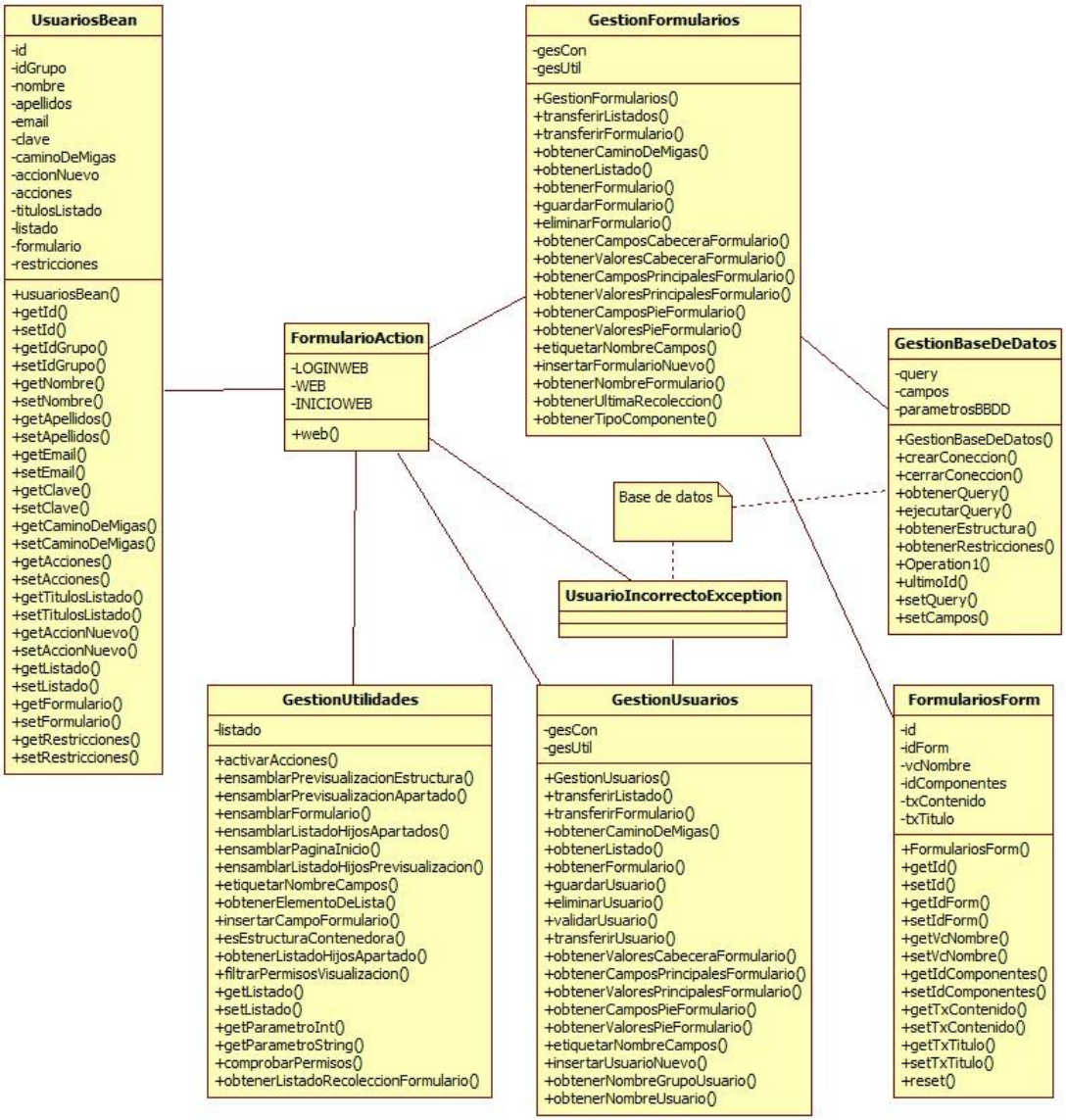


Figura 4.14: Envío de datos a través de un formulario de la web generada.

4.4.7. Clases utilizadas en las diferentes secciones.

Hay otras clases muy similares a las clases de la sección Estructuras (GestionEstructuras, EstructurasBean y EstructurasForm) que también tienen relación con las clases AdministracionAction, EdicionAction y GuardarAction mencionadas en los apartados *4.4.2 Administración de las secciones, 4.4.3 Edición de las secciones* y *4.4.4 Guardar el elemento de una sección*. La diferencia es que en vez de hacer referencia a las Estructuras hacen referencia a: GruposEstructuras, Apartados, GruposUsuarios, Usuarios, GruposPermisos, Permisos, Apariencias, Formularios y RecoleccionFormularios. Estas relaciones de las clases están pensadas para responder a las peticiones que se puedan realizar a través de las diferentes secciones del CMS. Teniendo el mismo procedimiento que con las Estructuras. En la Figura 4.15 solamente se muestra la definición de las clases con la nomenclatura Gestion en el nombre ya que las que tienen la nomenclatura Bean y Form solamente definen un listado de datos sin ninguna funcionalidad especial.

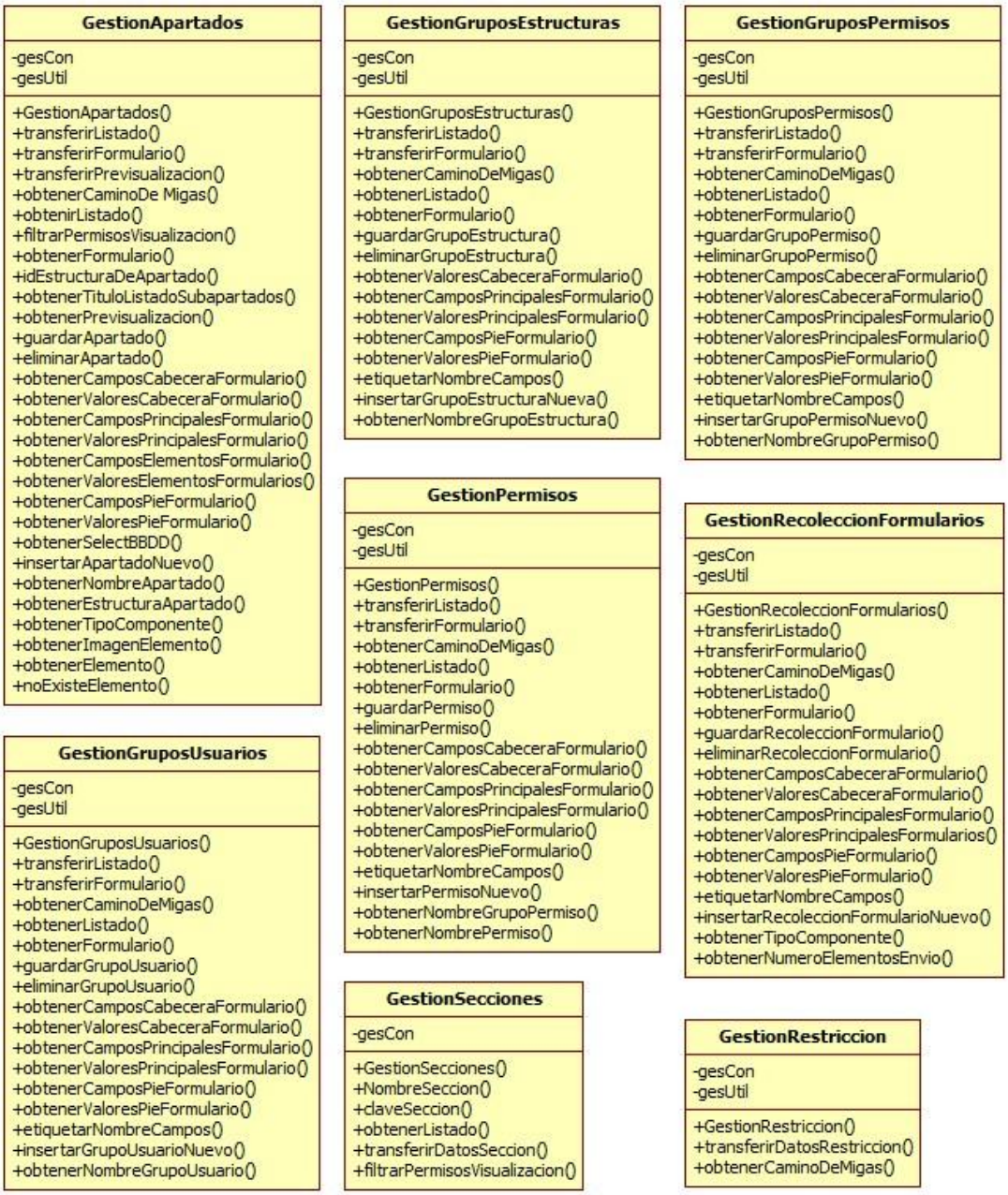


Figura 4.15: Clases Gestion

5. Diseño

En este capítulo se muestra el diseño de la aplicación web que se va a implementar, mediante los diagramas de secuencia, el diseño de las páginas webs que forman la aplicación, el mapa web y el diseño de la base de datos.

5.1. Diagrama de secuencia

Los diagramas de secuencia mostrados en este apartado describen las interacciones que se efectúan cuando un usuario realiza una petición a la aplicación, activando un objeto del tipo Action que se encarga de identificar el tipo de petición que se ha efectuado solicitar los datos necesarios a las clases de la capa Modelo y enviarlos a la capa Vista para que el usuario pueda ver el resultado.

5.1.1. Validación del usuario

Cuando el usuario realiza una identificación de login y contraseña se activa el objeto ValidacionAction, el cual recoge los datos del formulario que ha utilizado el usuario y manda comprobar a los objetos de la capa Modelo comprobar si estos datos se corresponden a un usuario existente. Si la identificación se cumple correctamente se guarda el identificador del grupo de usuario al que pertenece para que pueda seguir navegando por la aplicación con los permisos de acceso y restricciones que le pertenece. Si por lo contrario no se cumpliera la identificación de usuario se crearía una excepción que provocaría que el usuario volviera al formulario de identificación indicándole que no ha superado la identificación. Ver el diagrama de secuencia de la figura 5.1.

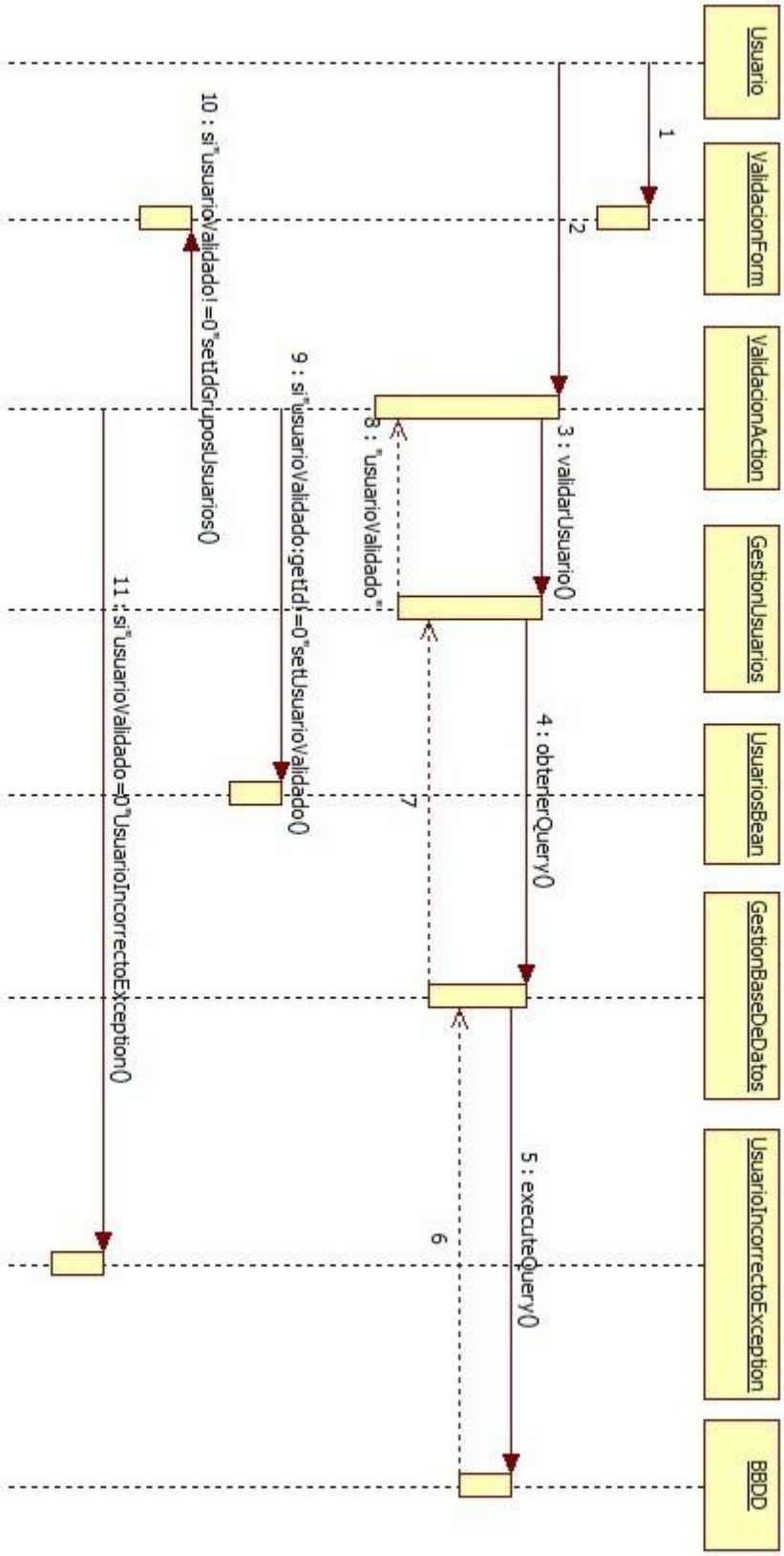


Figura 5.1: Diagrama de secuencia de la validación del usuario.

5.1.2. Administración de las secciones

Cuando un usuario accede a una sección de la administración lo primero que ve es un listado de los elementos que la componen. El objeto que se activa e interactúa con el resto de objetos necesarios para devolver como resultado el listado de elementos, es el objeto AdministracionAction. Antes de empezar a interactuar con el resto de objetos comprueba que el usuario ha realizado la identificación y que tiene los permisos necesarios para acceder a la sección. En el diagrama de la figura 5.2 se muestra las secuencias que se realizan cuando se accede a la sección de estructuras, cuando se accede a otra sección, las secuencias que se efectúan son las mismas pero sustituyendo la utilización de los objetos GestionEstructuras y EstructurasBean por las correspondientes a la sección.

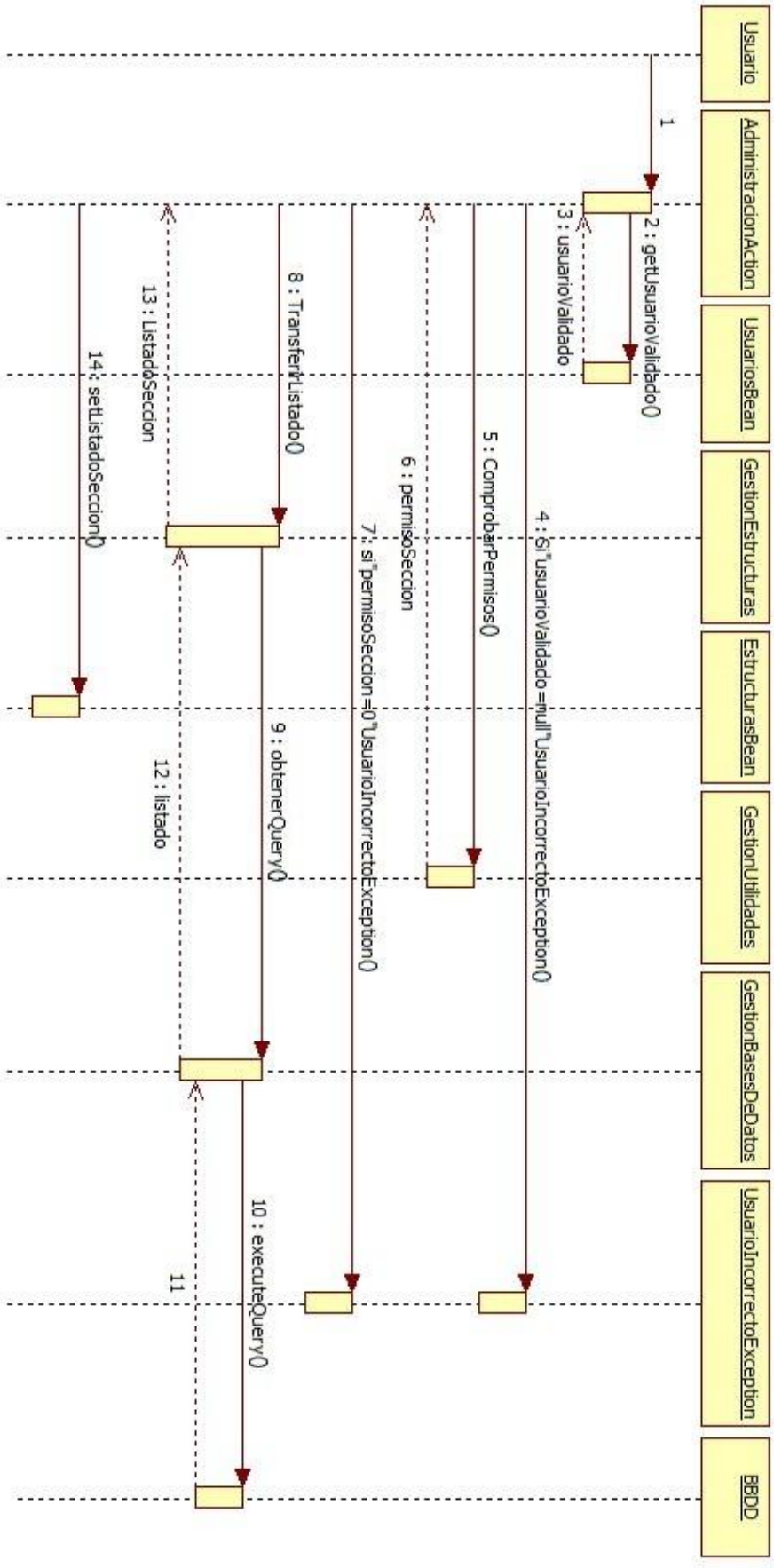


Figura 5.2: Diagrama de secuencia de la administración de secciones.

5.1.3. Edición de una sección

Al acceder al formulario que se utiliza para editar los elementos de cualquier sección se activa el objeto EdicionAction que interactúa con el resto de objetos necesarios para devolver como resultado el formulario. Antes de empezar a interactuar con el resto de objetos comprueba que el usuario ha realizado la identificación y que tiene los permisos necesarios para acceder al formulario. En el diagrama de la figura 5.3 se muestra las secuencias que se realizan cuando se accede al formulario de edición de las estructuras, cuando se accede al formulario de edición de otra sección, las secuencias que se efectúan son las mismas pero sustituyendo la utilización de los objetos GestionEstructuras y EstructurasBean por las correspondientes a la sección. Además de devolver el formulario de edición, este objeto también puede devolver una pre-visualización de un elemento o borrarlo de la base de datos.

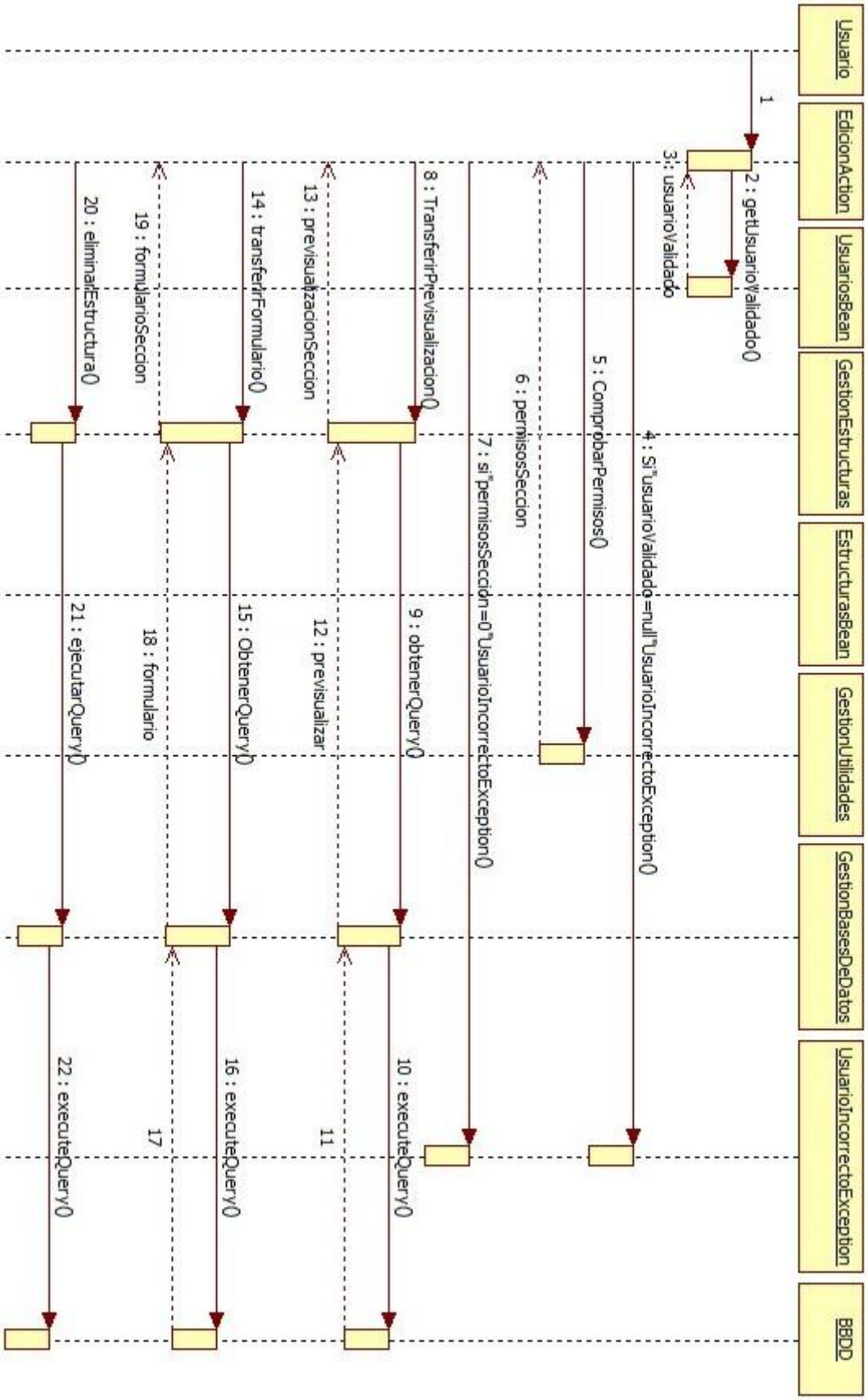


Figura 5.3: Diagrama de secuencia de la edición de una sección.

5.1.4. Guardar una sección

Cuando se guarda el elemento de una sección a través del formulario mencionado en el apartado anterior se activa el objeto GuardarAction que interactúa con el resto de objetos necesarios para guardar en la base de datos la información enviada a través del formulario de edición. Antes de empezar a interactuar con el resto de objetos comprueba que el usuario ha realizado la identificación y que tiene los permisos necesarios para acceder al formulario. En el diagrama de la figura 5.4 se muestra las secuencias que se realizan cuando se envía un formulario de edición de las estructuras, cuando se envía un formulario de edición de otra sección, las secuencias que se efectúan son las mismas pero sustituyendo la utilización de los objetos GestionEstructuras y EstructurasBean por las correspondientes a la sección.

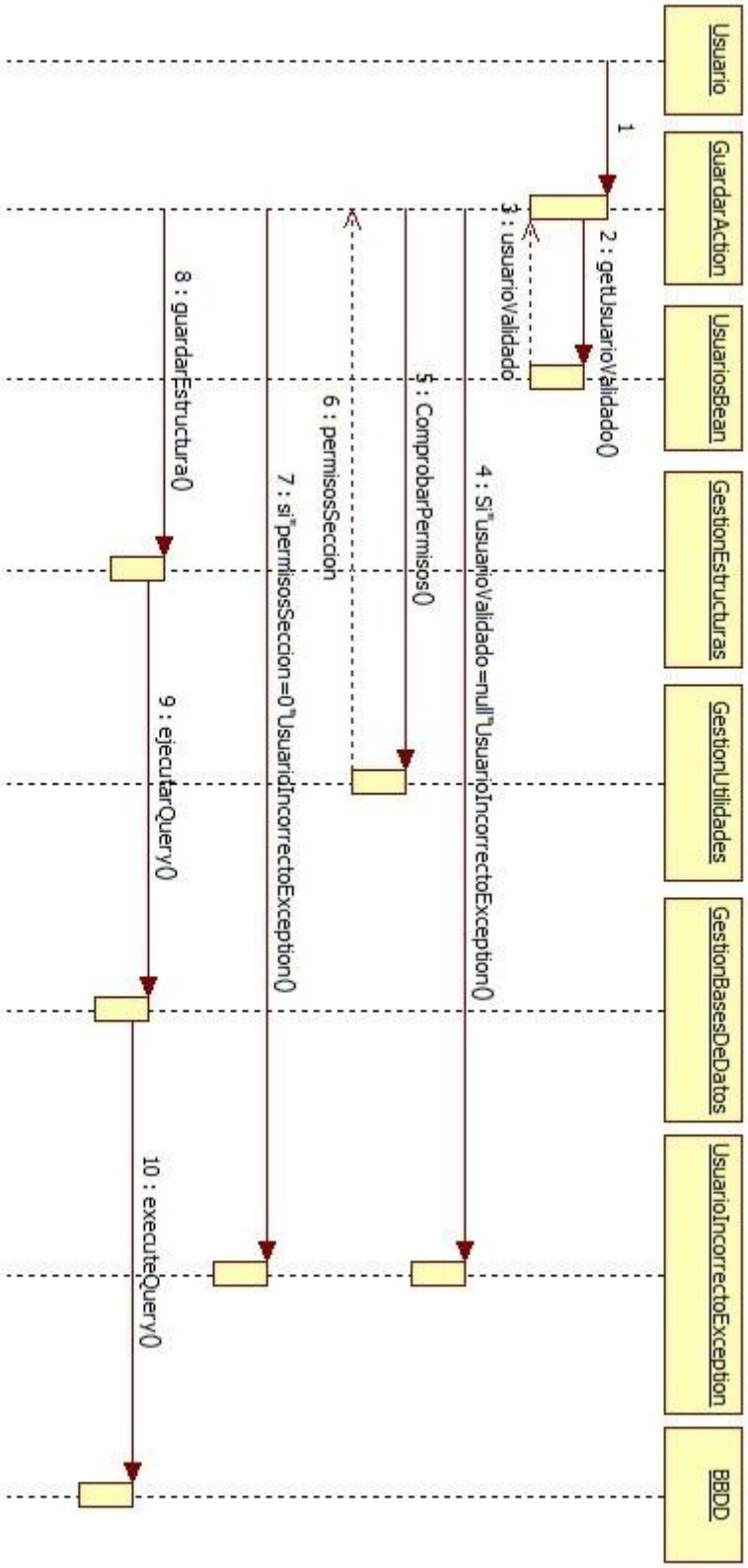


Figura 5.4: Diagrama de secuencia del guardado de una sección.

5.1.5. Visualización de la web generada por el CMS

En el momento que se accede a un apartado de la web generada por la aplicación se activa el objeto WebAction que interactúa con el resto de objetos necesarios para devolver el contenido, la estructura del apartado y la apariencia gráfica definidos desde la administración. Antes de empezar a interactuar con el resto de objetos comprueba que el usuario ha realizado la identificación para comprobar si puede acceder al apartado, en el caso de que no la haya realizado se le otorga los permisos de un usuario anónimo, el cual solamente puede acceder a los apartados que son totalmente públicos para todos los usuarios. En el diagrama de la figura 5.5 se muestra las secuencias que se realizan cuando se accede a un apartado de la web.

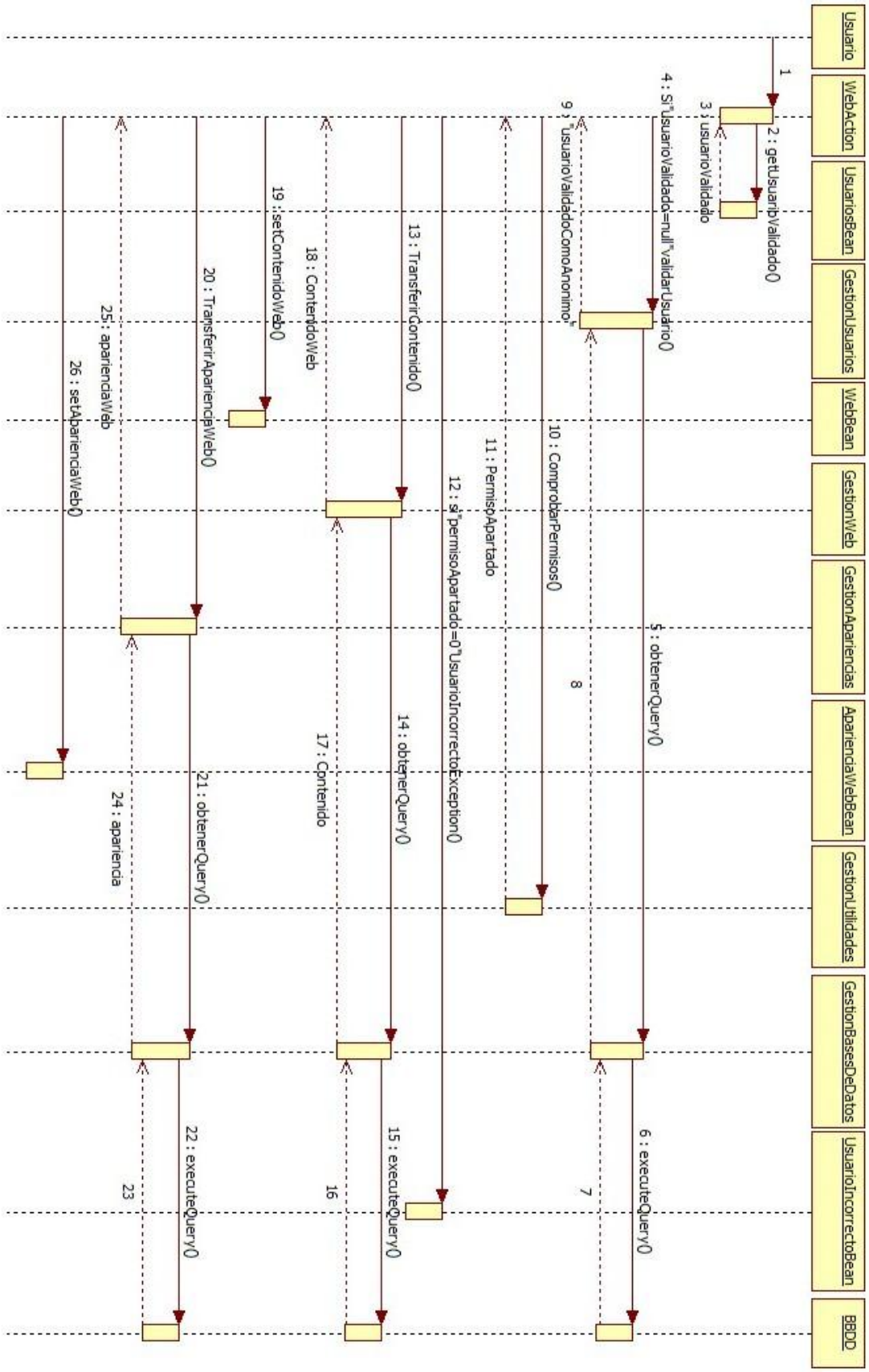


Figura 5.5: Diagrama de secuencia de la visualización de la web generada por el CMS.

5.1.6. Envío de un formulario a través de la web generada

Si se utiliza un formulario de la web generado por la aplicación se activa el objeto FormularioAction que interactúa con el resto de objetos necesarios para guardar los datos enviados a través del formulario. Antes de empezar a interactuar con el resto de objetos comprueba que el usuario ha realizado la identificación para comprobar si puede utilizar el formulario, en el caso de que no la haya realizado se le otorga los permisos de un usuario anónimo, el cual solamente puede utilizar los formularios de la web que son totalmente públicos para todos los usuarios. En el diagrama de la figura 5.6 se muestra las secuencias que se utiliza un formulario de la web.

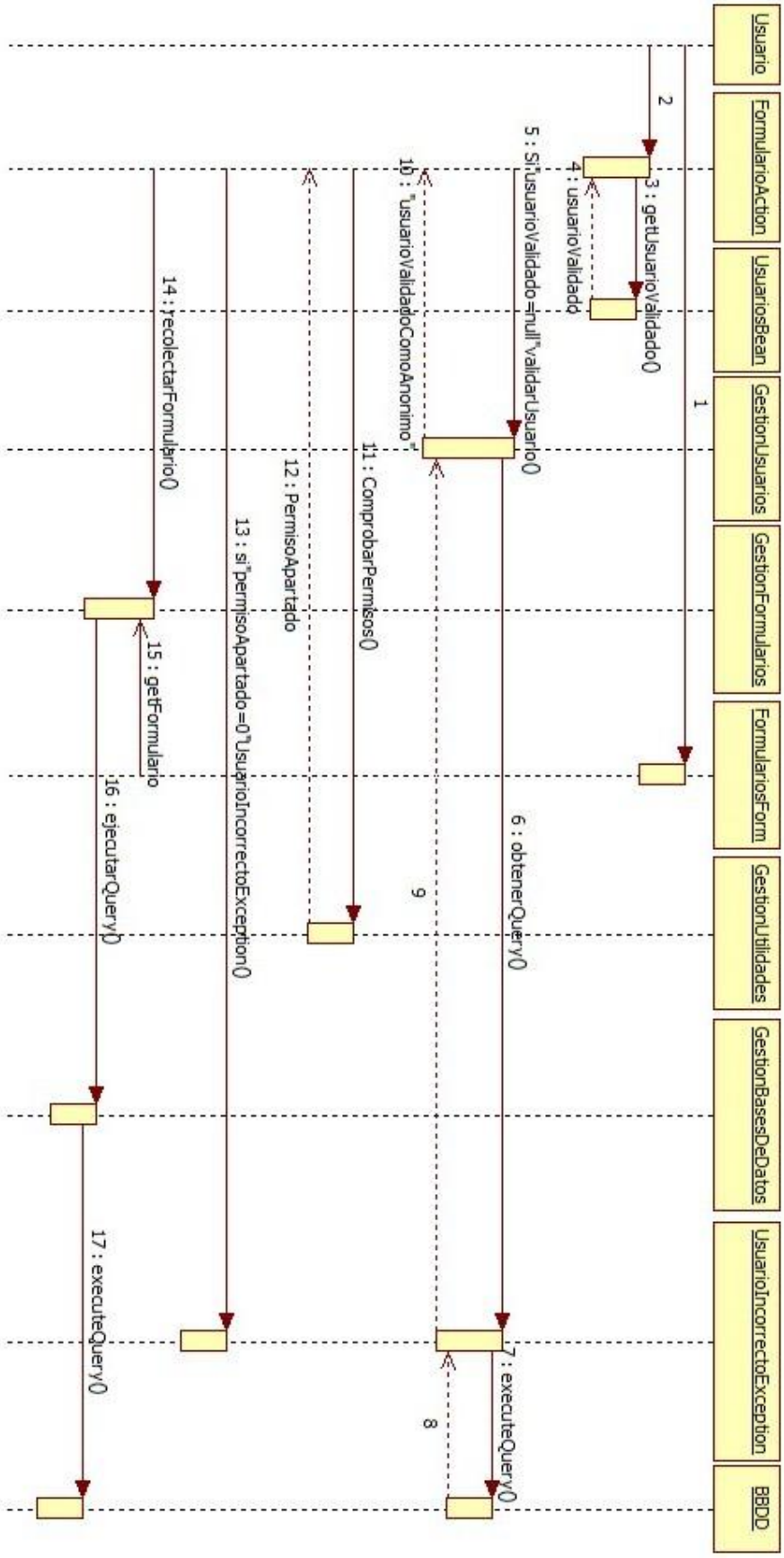


Figura 5.6: Diagrama de secuencia del envío de un formulario a través de la web generada.

5.2. Diseño de las páginas web

Al diseñarse la aplicación pensada en que sea fácil añadir nuevas secciones, todas las páginas web del CMS de las webs generadas tienen la misma estructura mostrada en la figura 5.7.

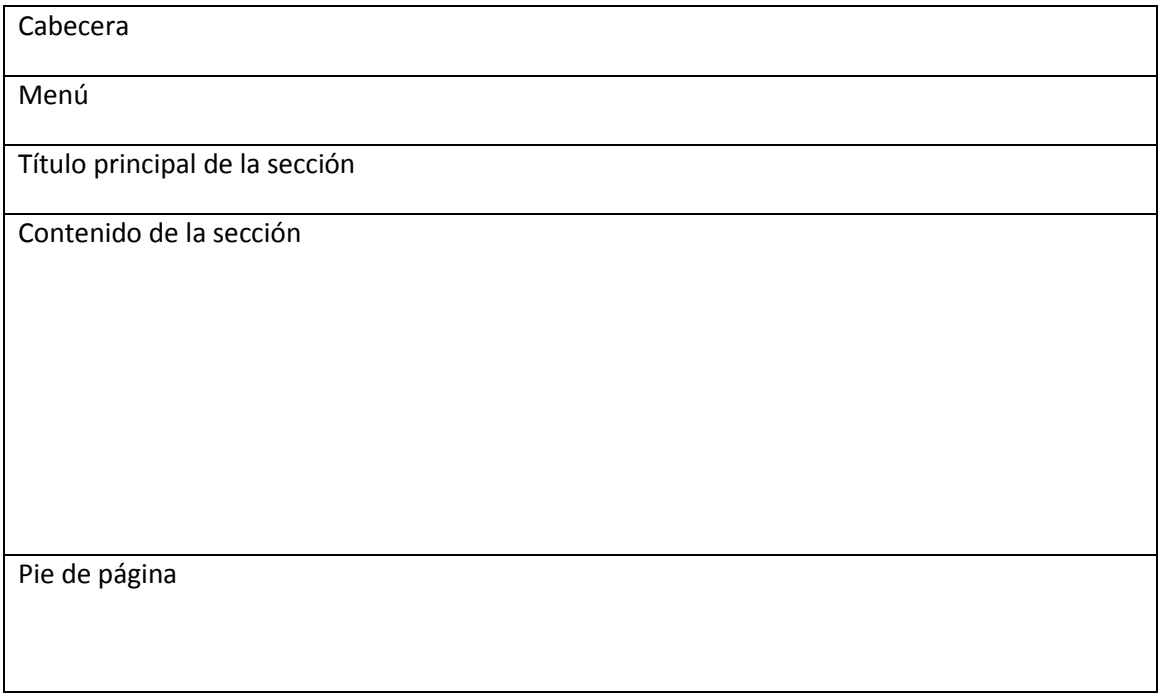


Figura 5.7: Diseño de las páginas web.

Dentro del CMS:

- En la cabecera se incluirá el título que se le ha dado a la aplicación: GCMS, Generador de gestores de contenido.
- En el menú se mostrará un listado de opciones con las diferentes secciones del CMS.
- En el título principal de la sección se mostrará la sección donde se acceda.
- El contenido podrá mostrará todas las opciones que ofrece la sección, como por ejemplo, un listado de los elementos que contiene, formulario para modificar los valores de un elemento de la sección o la pre-visualización de un elemento.
- En el pie de página se muestran referencias de la aplicación: links, direcciones, etc.

En la web generada:

- En la cabecera se incluirá el título que se le ha puesto desde el CMS.
- En el menú se mostrará un listado de opciones con los diferentes apartados de la web.
- En el título principal de la sección se mostrará el apartado donde se acceda.
- El contenido podrá mostrará el cuerpo del apartado definido desde el CMS mostrando su contenido y la estructura definida.
- En el pie de página se muestran referencias de la aplicación: links, direcciones, etc.

5.3. Mapa web

En la figura 5.8 se lista las opciones que contiene el CMS y las acciones que se pueden realizar.

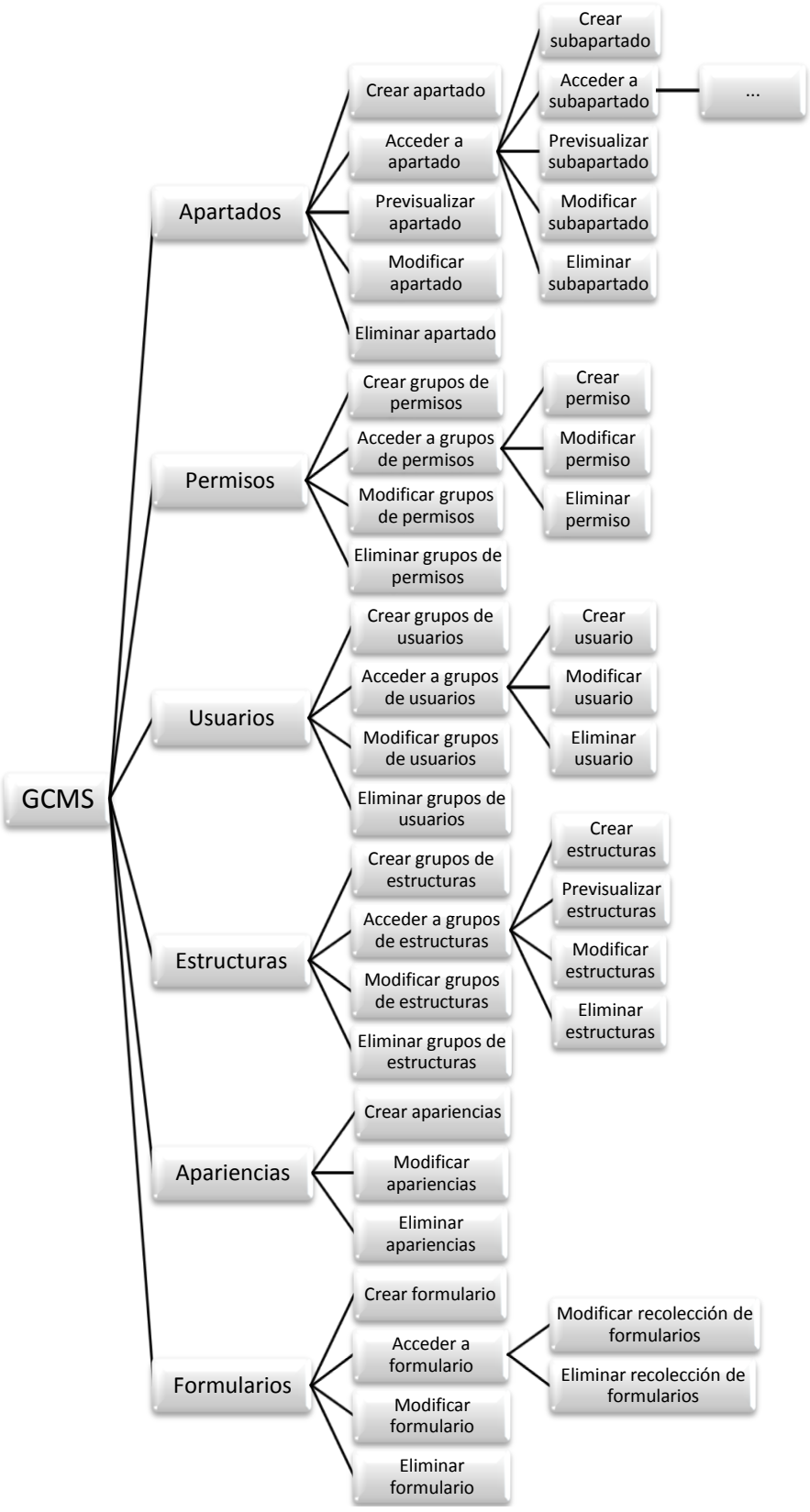


Figura 5.8: Mapa web del CMS.

5.4. Diseño de la Base de datos

Este apartado presenta las tablas y la estructura de la base de datos utilizada por el CMS que se va a desarrollar. Explicando las diferentes tablas y su mostrando la relación entre ellas.

5.4.1. Tablas de la base de datos.

El grupo de tablas más importantes almacenan datos de los elementos de las diferentes secciones:

Secciones: almacena información general de las diferentes secciones que se encuentran en el CMS (Apartados, Estructuras, Usuarios, Permisos, Apariencias y Formularios).

Gruposestructuras: almacena los datos de todos los Grupos de estructuras.

Estructuras: almacena los datos de todas las Estructuras, menos los datos de sus Componentes.

Componentes: almacena los datos de todos los Componentes incluyendo a que Estructura pentenecen.

Apartados: almacena los datos de todos los Apartados, menos los valores que están definidos por los Componentes

Elementos: almacena los datos de todos los Elementos que representa dato guardado desde un Apartado definido por los Componentes desde la Estructura del tipo de Apartado. Todos los elementos están relacionados con un Apartado, y un Componente.

Gruposusuarios: almacena los datos de todos los Grupos de usuarios.

Usuarios: almacena los datos de todos los Usuarios.

Grupospermisos: almacena los datos de todos los Grupos de Permisos.

Permisos: almacena los datos de todos los Permisos.

Apariencias: almacena los datos de todas las Apariencias.

Formularios: almacena los datos de todos los Formularios.

Recoleccionformularios: almacena los datos de todas las Recolecciones de formularios.

Al realizarse la aplicación pensando en que sea fácil incluir nuevas secciones teniendo que programar lo mínimo posible, se ha realizará la programación para que todas las páginas de listados y formularios se generen automáticamente al leer los campos de las tablas, teniendo en cuenta su nomenclaturas. Por este motivo hay tablas que almacenan datos mostrados en listados de un formulario, por ejemplo: si/no, derecha/centro/izquierda. Estas tablas son: **Escampoformulario**, **Esobligatorio**, **Tipocentrado**, **Tiposalineaciones**, **Tiposcomponentes**, **Tiposdireccionorden**, **Tiposordensubapartados**, **Tipospermisos**, T**iposrecuadrocontenido** y **Tipossino**.

5.4.2. Diagrama de la base de datos

En la Figura 5.9 se muestra las relaciones existentes entre las tablas de la base de datos.

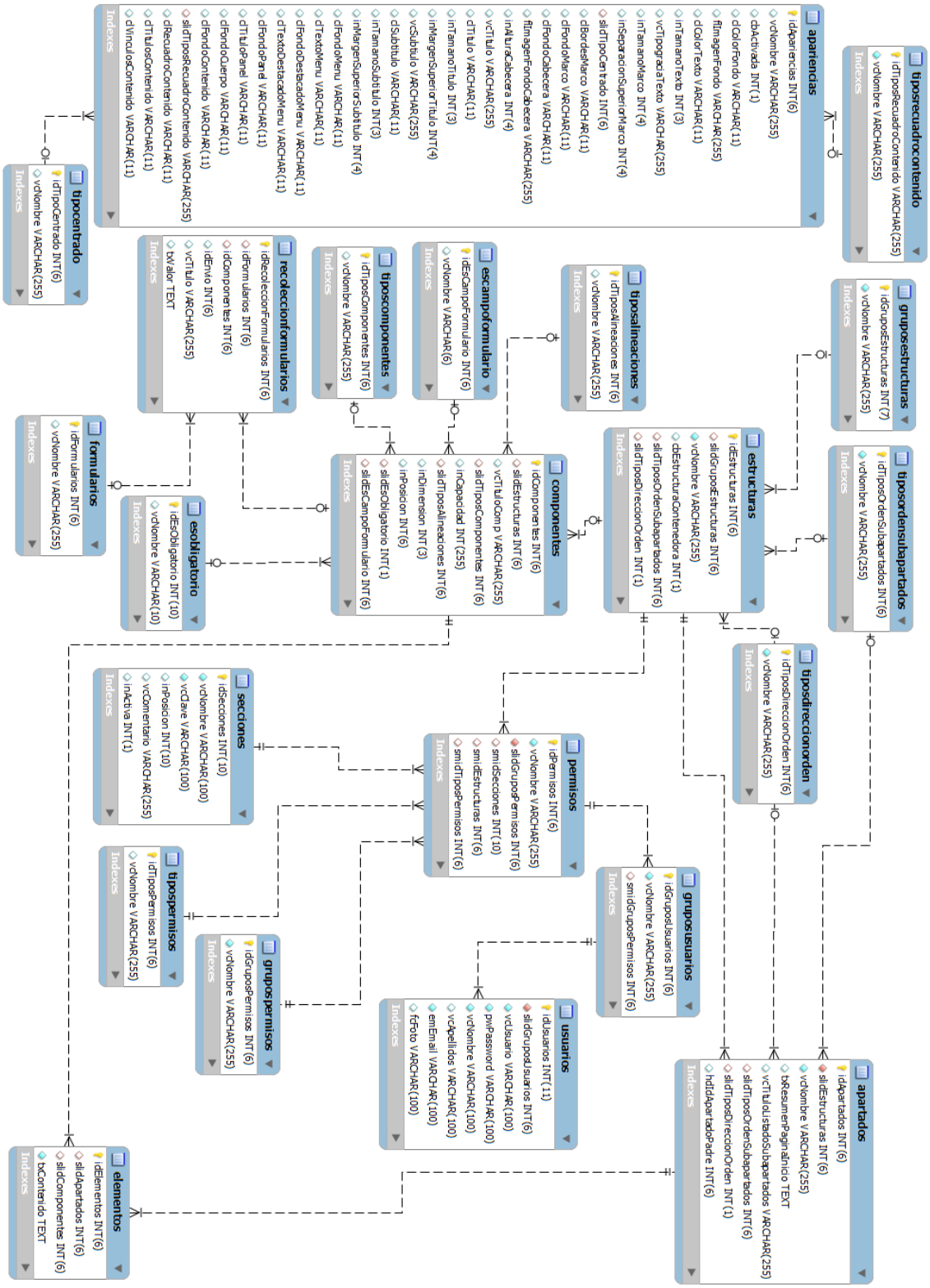


Figura 5.9: Diseño de la base de datos.

6. Implementación

En este capítulo se describe la implementación de la aplicación, mencionando los lenguajes de programación y las tecnologías utilizadas, las metodologías propias de los desarrolladores, los problemas que se han encontrado durante el desarrollo del proyecto, una explicación de la utilización de CMS que se ha desarrollado incorporando capturas de pantalla de las páginas web y una batería de pruebas realizadas sobre el proyecto desarrollado.

6.1. Lenguajes de programación y tecnologías utilizadas

Para la realización del CMS se han utilizado los las siguientes tecnologías explicadas en el capítulo 3:

- **Lenguajes de programación:** Html, JSP, Java, Struts, Css y Jquery.
- **Servidores que necesita la aplicación para funcionar**: Mysql y Tomcat.
- **Programas que han ayudado a editar la programación:** El editor Java Netbeans, el pluging Firebug de Firefox que ayuda a ver la programación y sus interacciones de css y javascript de una página web desde el navegador y editor de bases de datos Navicat.
- **Metodologías que se han utilizado:** Programación orientada a objetos y la estructuración de capas MVC.

6.2. Metodologías propias del desarrollador

En este apartado se explica las metodologías definidas por los desarrolladores que se han seguido a la hora de programar el código de la aplicación, tales como: las clases utilizadas, metodologías de seguridad de acceso a las páginas web, nomenclaturas utilizadas en el análisis y desarrollo de la aplicación y los pasos que se podrán seguir en un futuro para incluir fácilmente una nueva sección dentro de la aplicación.

6.2.1. Los diferentes tipos de clases utilizadas

Con el objetivo de satisfacer la metodología MVC impuesta por el framework Struts, en la realización de la programación se ha dividido las clases definidas en varios grupos, ofreciendo cada grupo un tipo de funcionalidad. Entre estos grupos tenemos: Las clases Action de la capa Controlador, las Clases Gestion de la capa Modelo que son las encargadas de crear la información que se presentará en la web y las páginas JSP de la capa Vista. Para transportar los datos utilizados en la aplicación de una capa a otra se utilizan las clases Form y las clases Bean que son clases tipo JavaBeans con la diferencia que las Form están definidas por el framework Struts y se encargan de recolectar automáticamente los datos enviados a través de los formularios, ahorrando al programador tener que realizar esta parte del código.

6.2.2. Metodología de seguridad

Cuando un usuario accede a la aplicación pasa un proceso de identificación para averiguar a qué grupo pertenece y los permisos y restricciones que tiene, guardándolo en una variable durante el tiempo que está navegando por la aplicación. Cada vez que accede a una sección primero se comprueba si ha pasado el proceso de identificación y después si tiene los suficientes permisos como para acceder a la sección en particular, denegándole el acceso si no los tiene.

6.2.3. Los diferentes tipos de funciones y el orden dentro del código

Dentro de las clases de la capa Modelo que como se ha comentado es donde se centra la mayor parte de la programación de los desarrolladores se suele seguir una nomenclatura y orden con los nombres de los métodos consiguiendo así una mejor estructura de la programación y mejor claridad a la hora de saber que hace cada método:

- Los métodos que comienzan por la nomenclatura *transferir* son los que se llaman desde la capa Controlador y se encargan de montar a groso modo mediante otros métodos toda la información solicitada necesaria para montar una página web y transferirla en una clase Bean a la capa Controlador.
- Los métodos que comienzan por la nomenclatura obtener se encargan de obtener información más específica una parte de la web en concreto. Son los métodos que utilizan los timo transferir para recolectar toda la información que contiene una página.
- Los métodos que comienzan con la nomenclatura guardar y eliminar son las funciones que dan la orden de modificar el valor del contenido de la base de datos.
- Para el resto de métodos se utilizan otras nomenclaturas de menor relevancia.

6.2.4. Las nomenclaturas de los campos de la base de datos

En la definición de los campos de las tablas también se sigue una nomenclatura que indica que tipo de dato es cada campo. Se han programado unas clases que generan formularios dinámicamente utilizando estas nomenclaturas, así se consigue crear nuevos formularios solamente definiéndolos en la base de datos y aprovechando el mismo código de programación. Estas nomenclaturas son:

- id: identificador del elemento que en el formulario se incluye como un campo oculto.
- vc: campo de texto.
- pw: campo con formato codificado donde se introduce una contraseña.
- tx: area de texto de gran tamaño.
- in: campo de texto que contiene un número.
- cb: campo de confirmación conocido como checkbox.
- rd: campo de selección de una opción conocido como radiobutton.

- slid: campo selector de una opción a escoger entre los elementos de otra tabla indicada con el resto del nombre, conocido como select.
- smid: campo selector de más de una opción a escoger entre los elementos de otra tabla indicada con el resto del nombre, conocido como select múltiple
- fl: campo donde se puede seleccionar una imagen.
- ar: campo donde se puede seleccionar un fichero.
- cl: campo donde se indica un color.

6.2.5. Las clases GestionBasesDeDatos y GestionUtilidades

Estás clases son llamadas por la mayoría de las clases y se utilizan para desarrollar acciones muy comunes evitando así repetir en exceso código de programación. En el caso de la Clase GestionBasesDeDatos se encarga de realizar todos los pasos necesarios para conectarse con la base de datos y ejecutar la instrucción que le indica la clase que la ha llamado devolviéndole el resultado obtenido. La clase GestionUtilidades entre otras cosas se encarga de crear los formularios teniendo en cuenta la nomenclatura de la base de datos, de crear el contenido de la página web teniendo en cuenta la definición creada en las estructuras y de comprobar si un usuario tiene los permisos suficientes como para acceder a una sección de la aplicación.

6.2.6. Pasos a seguir para crear fácilmente una nueva sección

Uno de los objetivos de la realización del proyecto era programar la aplicación de forma que fuera fácil añadir nuevas secciones que puedan ofrecer nuevas funcionalidades para definir cada vez un tipo de web más completa. Con este objetivo se ha listado una serie de pasos para poder generar una nueva sección genérica (ver listados, crear elemento, modificar elemento, borrar elemento) teniéndose que programar posteriormente las funcionalidades específicas de casa sección. Estos pasos son:

1. En la Base de datos:
    - Añadir el nombre de la nueva sección en la tabla secciones de esta manera aparecerá en el menú de la aplicación.
    - Crear una nueva tabla referente a la sección definiendo todos sus elementos teniendo en cuenta las nomenclaturas de tipo de elemento definida anteriormente.
2. En el fichero de configuración struts-config.xml:
    - Dentro del tag <global-forwards> añadir el <forward> que hace referencia a la nueva sección para que cuando se clique en el menú ha esta sección redirija correctamente.
    - Dentro del tag <action-mappings>: añadir el <action> que referencie a la clase "AdministracionAction" incluyendo el parámetro “seccionNombreDeLaNuevaSeccion”, con esta declaración se define que cuando se realice una petición web relacionada con acceder a la nueva sección se ejecutará el método seccionNombreDeLaNuevaSeccion de la clase

AdminstracionAction de la capa Controlador que hace referencia a esta petición.

3. En la clase AdministracionAction:
    - Añadir el método seccionNombreDeLaNuevaseccion que se encargará de gestionar la petición al acceso de la nueva sección. Pedir los datos necesarios para crear el listado de la sección y enviárselos a la capa Vista para que lo imprima por pantalla.
4. Crear las clases NombreDeLaNuevaSeccionForm y NombreDeLaNuevaSeccionBean definiendo los datos que se quieren transportar de una capa a otra de la aplicación. En el caso de la clase tipo Form lo datos deben coincidir con los campos definidos en la tabla de la sección de la base de datos puesto que recogerá automáticamente los formularios creados a raíz de la definición de la base de datos. Y en el caso de la clase tipo Bean se debe definir los datos que se quieren transferir a la capa Vista necesarios para crear la página web que verá el usuario.
5. Crear la clase GestionNombreDeLaNuevaSeccion está es la clase que realizará todas las acciones a medida para satisfacer las funcionalidades de la nueva sección.
6. En el fichero de configuración struts-config.xml:
    - Dentro del tag <form-beans> añadir la definición <form-bean> definir la clase NombreDeLaNuevaSeccionForm
    - Dentro del tag <action-mappings> vincular la clase NombreDeLaNuevaSeccionForm con la clase guardarAction que se encargará de recoger los valores del formulario relacionado con la nueva sección y enviárselos a la capa Modelo para que los guarde en la base de datos.
7. En la clase EdicionAction:
    - Añadir el método NombreDeLaNuevaSeccion que se encargará de gestionar la petición al acceso de la nueva sección. Pedir los datos necesarios para crear el formulario de la sección y enviárselos a la capa Vista para que lo imprima por pantalla.
8. En el fichero de la capa vista edicion.jsp:
    - Añadir el tag <html:javascript formName="NombreDeLaNuevaSeccionForm" /> que se encarga de incluir la codificación de javascript que validará los datos que se envían a través de los formularios de la nueva sección tal y como se han definido en los ficheros de configuración validation.xml y validator-rules.xml.
    - Añadir un nuevo bloque de formulario con la cabecera <html:form action="admin/guardarNombreDeLaNuevaSeccion" method="post" enctype="multipart/form-data" onsubmit="return validateNombreDeLaNuevaSeccionForm(this);"> esta cabecera solicitará una petición a la clase GuardarAction enviandole los datos introducidos en el formulario a través de la clase NombreDeLaNuevaSeccionForm.
9. En el fichero de configuración ApplicationResource se añadirán todas las etiquetas que hacen referencia a traducciones de los nombres de los campos de la base de datos referentes a los nombre de los campos de los formularios. De esta manera si se quiere cambiar un nombre se hará en este fichero. O si se quiere tener la web en varios

idiomas se tendrá un fichero de este tipo por cada idioma leyendo las traducciones que interesen.

10. En los ficheros validation.xml y ApplicationResource.properties se añadiran las reglas de validación de los campos de los formularios que se quieran validar y los mensajes de error en caso de que no pasen dichas reglas de validación.
11. En el fichero GuardarAction se añadirá el método NombreDeLaNuevaSeccion que se encargará de gestionar la petición del envío de los datos de los formularios y enviárselos a la capa Modelo para que los gestione correctamente guardándolos en la base de datos si es necesario.

6.3. Problemas en el desarrollo del proyecto

El primer problema que se ha encontrado en la realización el proyecto está relacionado con el hecho de que no se tenían la experiencia suficiente en la programación web en Java y puesto que la metodología de este lenguaje es diferente al de otros lenguajes utilizados ha costado un esfuerzo superior al calculado aprender a desarrollar sobre la plataforma Struts. Y como suele pasar cuando se empieza a desarrollar por primera vez sobre una tecnología, se va aprendiendo a utilizar los conocimientos teóricos sobre la práctica, encontrándose en el caso de no haber enfocado el diseño de la implementación de la manera más óptima, provocando también invertir más tiempo del calculado en el desarrollo.

Otro problema que se ha tenido ha sido tener menos tiempo del deseado para poder dedicar al desarrollo de la aplicación a causa de horarios laborales.

Por último un problema muy común en la programación ha sido que al desarrollar código muy genérico utilizado con la idea de ser reaprovechado en muchas situaciones diferentes, se ha encontrado que en funcionalidades de la aplicación que funcionaban correctamente, al añadir una nueva modificación al código, ha traído como consecuencia que la funcionalidad que se daba por buena dejara de funcionar, teniendo que volver a modificarla para que fuera válida para las dos situaciones.

6.4. Utilización de la aplicación

En este apartado se muestra mediante imágenes todas las tareas que se pueden efectuar para la generación de una web a través del CMS que se ha desarrollado.

6.4.1. Identificación del usuario

La figura 6.1 muestra el formulario de autentificación que hay que validar para poder acceder a la administración del CMS.

Figura 6.1: Formulario de autentificación.

6.4.2. Administración de Estructuras

Cuando se accede a la sección de Estructuras lo primero que se ve es el listado de los Grupos de estructuras que contienen las diferentes estructuras definidas en el CMS Figura 6.2.

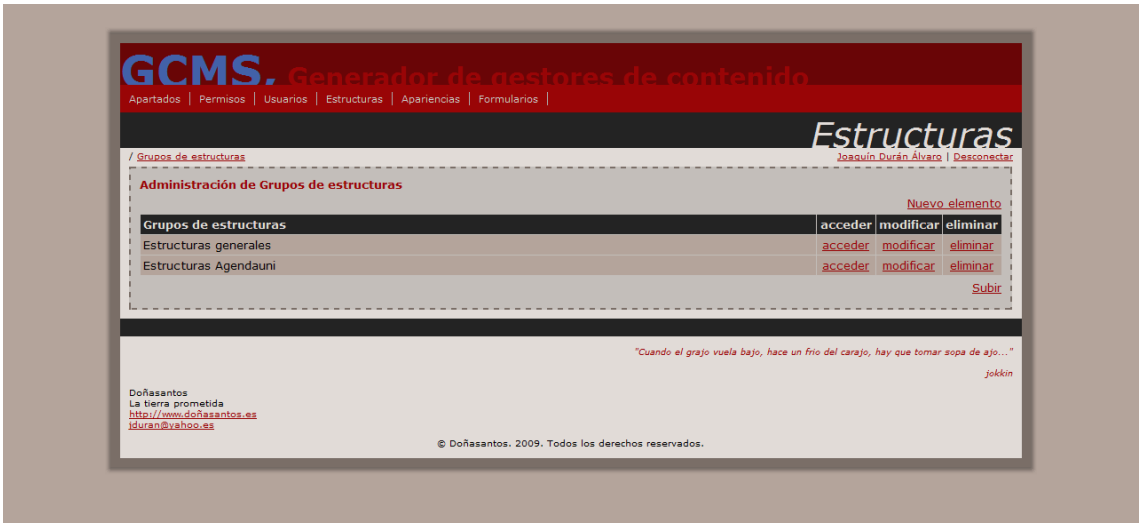


Figura 6.2: Listado de los Grupos de estructuras

Dando la opción de crear un nuevo grupo o modificar uno existente Figura 6.3, o de acceder al grupo para ver el listado de las Estructuras que contiene Figura 6.4.

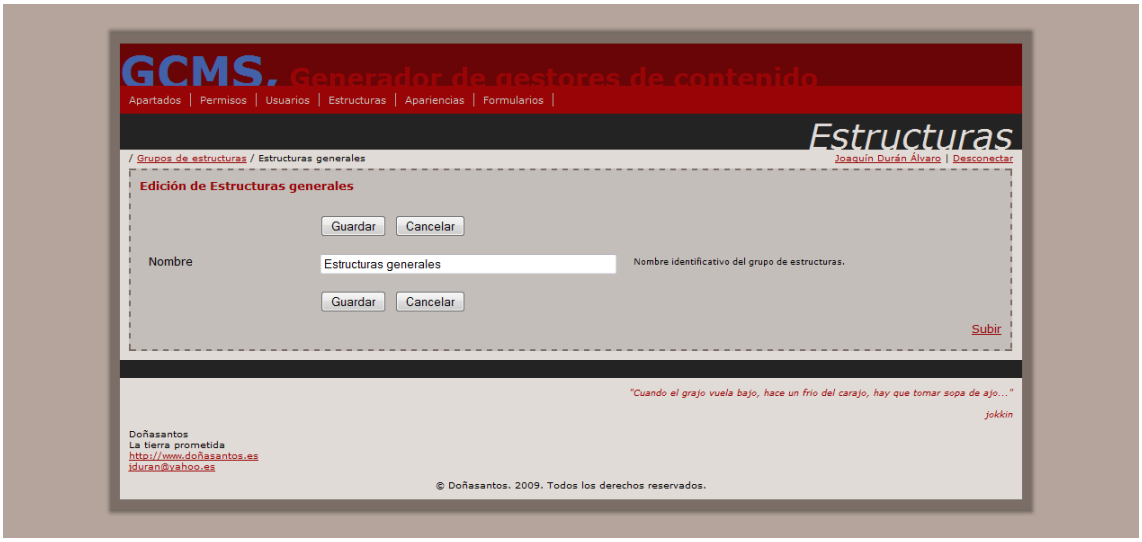


Figura 6.3: Edición de un Grupo de estructura

Figura 6.4: Listado de las estructuras pertenecientes a un Grupo de estructura.

Dentro de la edición de una estructura se pueden añadir, modificar y borrar los componentes que forman su esqueleto Figura 6.5.

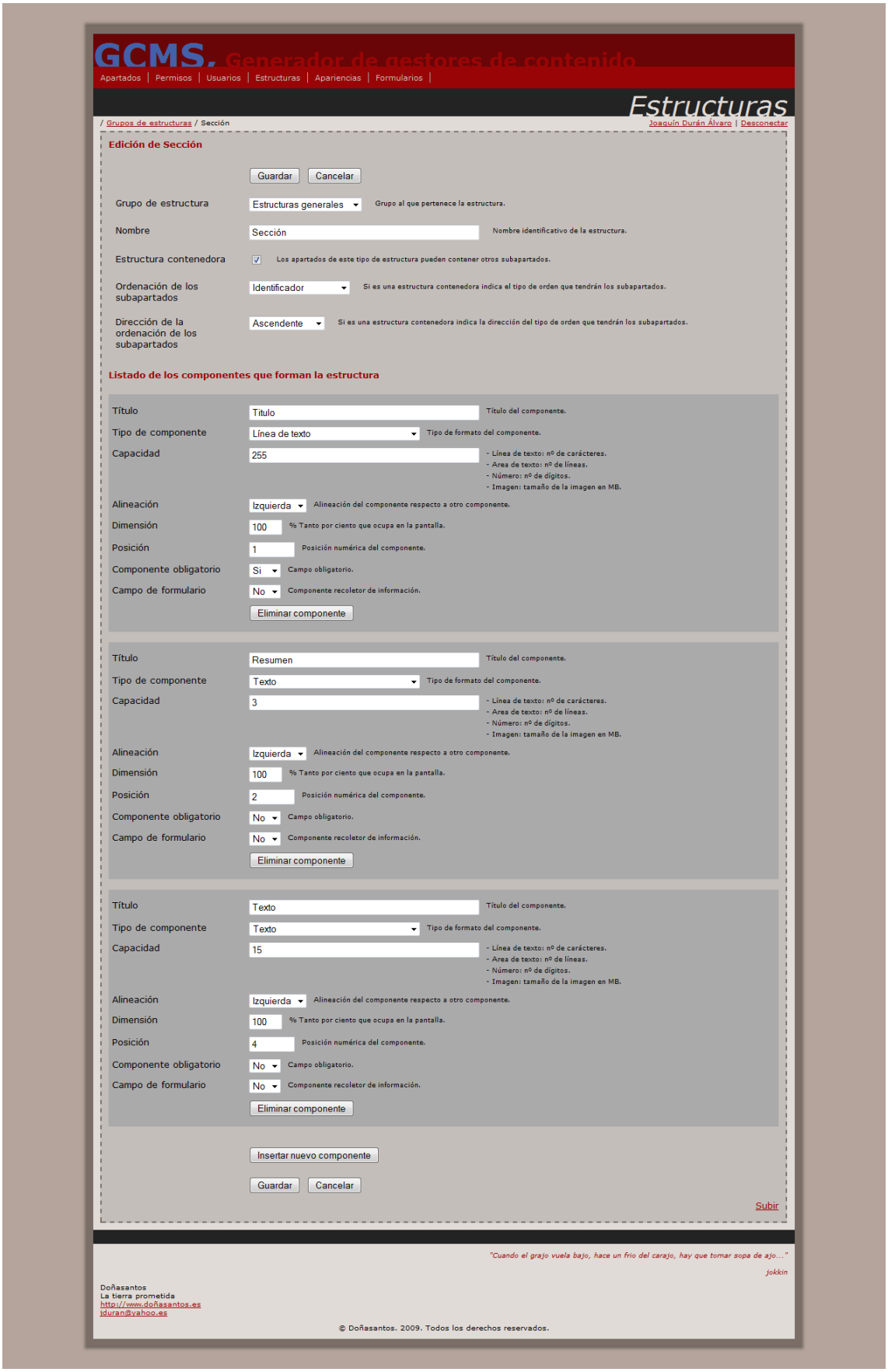


Figura 6.5: Edición de una Estructura.

Desde el listado de las estructuras también está la opción de pre-visualizarlas viendo una imagen orientativa de cómo quedará un apartado que utilice la estructura Figura 6.6.

Figura 6.6: Pre-visualización de una Estructura.

En la implementación del proyecto se asume que hay unas estructuras comunes que pueden ser utilizadas en la mayoría de webs que se creen a través del CMS, por eso ya se dejan creadas dentro de un Grupo de Estructuras llamado *Estructuras Generales,* estas estructuras son:

- **Sección:** contenedoras de otras estructuras, su única función es la de separar los contenidos de la web en diferentes secciones. Por ejemplo un apartado del tipo sección llamado noticias contendrá un conjunto de apartados del tipo noticia. Las secciones estarán compuestas por un título, una imagen, un resumen y un texto.
- **Noticia:** esta estructura está compuesta por un título, una imagen, un resumen y un texto.
- **Artículo:** esta estructura está compuesta por un título, una imagen, un resumen y un texto. Aunque está compuesto por los mismos componentes que una noticia, la distribución de estos por la pantalla es diferente.
- **Imagen:** los componentes que la forman son un título, una imagen y un comentario.

- **Agenda:** está pensada para contener estructuras de tipo evento que consta de un título, un resumen y un texto.
- **Evento:** Los eventos están formados por una fecha de evento, un título, una imagen, un resumen y un texto.
- **Formulario:** es una estructura pensada para definir un formulario desde el cual los usuarios que accedan podrán utilizarlo y guardar los datos en la aplicación. Está compuesto de los siguientes componentes de tipo texto: un título, un resumen, y texto y los componentes indicados como campos de formulario: Formulario, Nombre, Apellidos, Correo electrónico, Comentario, Botón de enviar
- **Galería Fotográfica:** pretende guardar fotos relacionadas con el pueblo y se compone de un título, un resumen, un texto y cinco fotos
- **Fichero:** los componentes que la forman son un título, un fichero y un comentario.

6.4.3. Administración de Apartados

Cuando se accede a la sección de apartados se ve el listado de Apartados que forma el contenido de la web que genera el CMS, figura 6.7. Estos Apartados se pueden crear o modificar como se ve en la Figura 6.8, pre-visualizar como en la Figura 6.9, eliminar y acceder dentro para ver el listado de apartados que contiene.

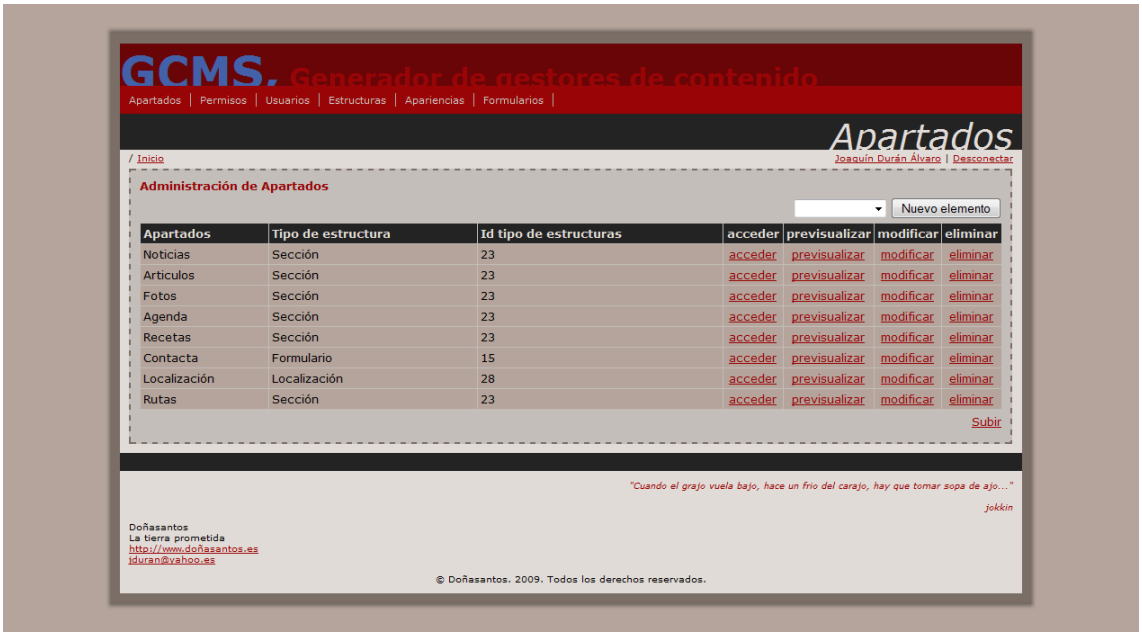


Figura 6.7: Listado de un grupo de apartados.

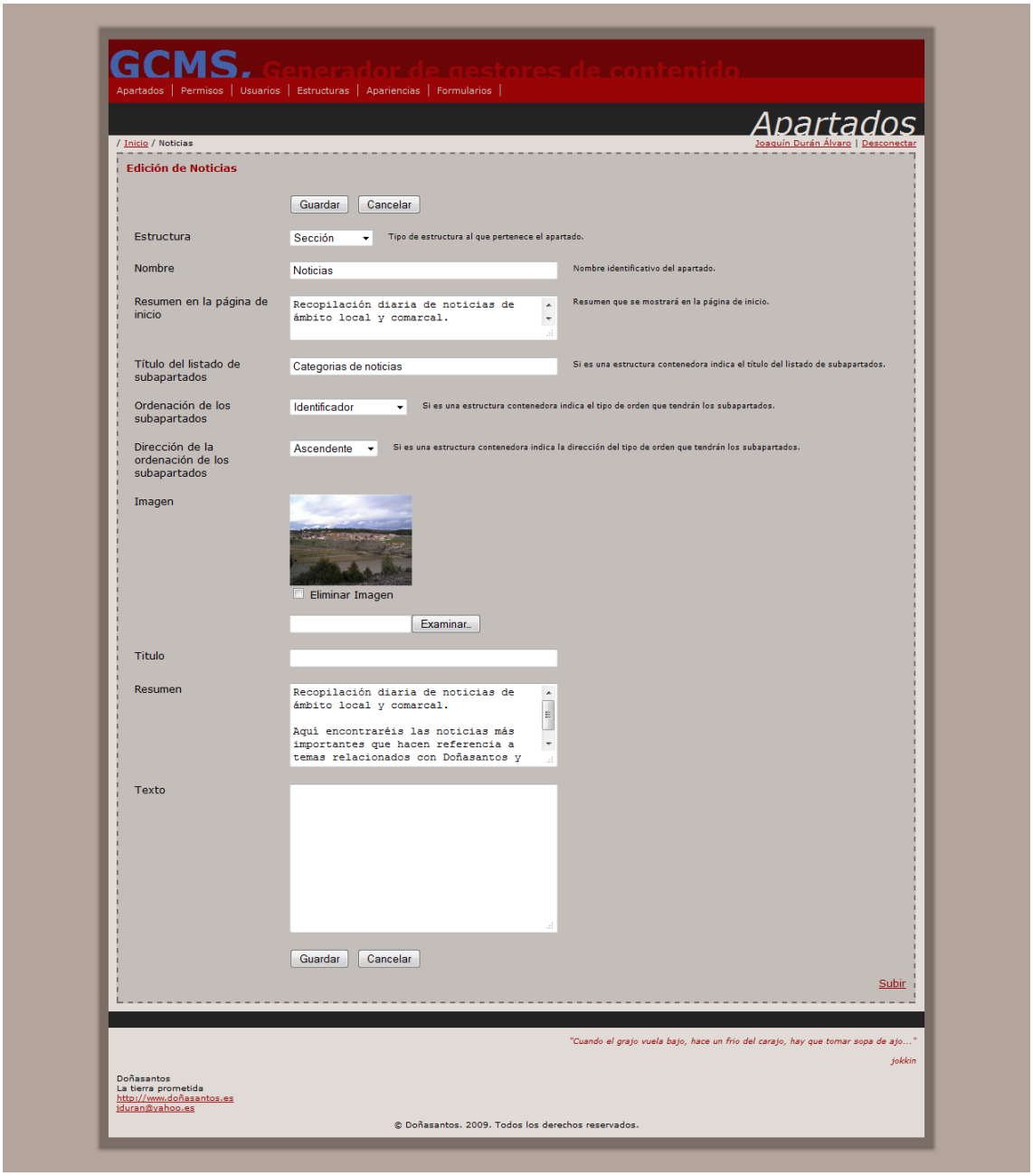


Figura 6.8: Edición de un Apartado.

Figura 6.9: Pre-visualización de un apartado.

6.4.4. Administración de Permisos

Cuando se accede a la sección de Permisos primero se ve el listado de los Grupos de permisos que contienen los diferentes permisos definidos Figura 6.10.

Dando la opción de crear un nuevo grupo o modificar uno existente Figura 6.11, o de acceder al grupo para ver el listado de las Permisos que contiene Figura 6.12.

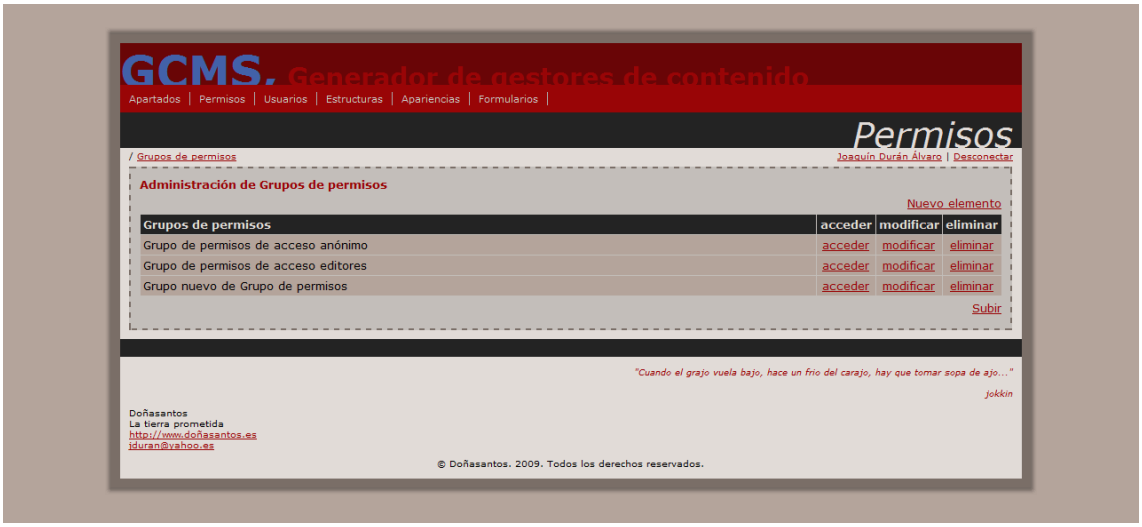


Figura 6.10: Listado de los Grupos de Permisos

Figrura 6.11: Edición de un Grupo de permisos.

Figura 6.12: Listado de un grupo de Permisos.

Los Permisos que se encuentran dentro de un Grupo de Permisos se pueden editar Figura 6.13 y eliminar. En la edición de un permiso se da a escoger las secciones de la administración y las Estructuras definidas y vincularlas a un tipo de restricción:

- **Visualización:** el usuario que tenga este tipo de restricción sobre una sección o apartado no podrá visualizarlos, ni siquiera en el menú.
- **Acceso:** el usuario que tenga este tipo de restricción sobre una sección o apartado podrá visualizarlos pero no podrá acceder a ellos.
- **Edición:** el usuario que tenga este tipo de restricción sobre una sección o apartado podrá visualizar y acceder pero no podrá modificar su contenido.

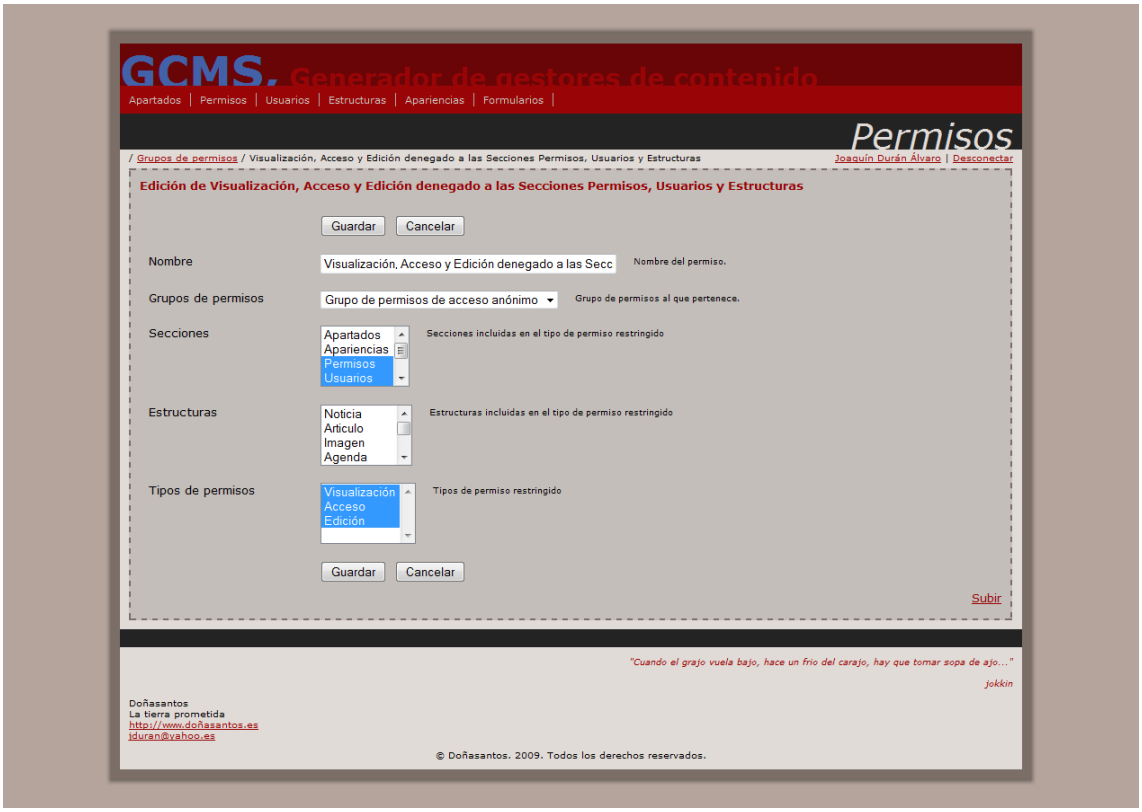


Figura 6.13: Edición de un Permiso.

Igual que en las Estructuras se asume que hay una serie de permisos comunes que pueden ser utilizados en la mayoría de webs que se creen a través del CMS, por eso ya se dejan creados estos permisos son:

- Permiso de administración: otorga a los usuarios unos permisos de acceso completo tanto a la administración como a la web, pudiendo definir las estructuras, las apariencias, los usuarios, los permisos, los formularios y editar el contenido de los apartados.
- Permiso de editor: ofrece a los usuarios acceso completo a la web pero dentro de la administración solamente tendrán permisos para acceder a la zona de edición de apartados, formularios y la apariencia de la web quedándole restringido el acceso a las estructuras, usuarios y permisos.
- Permiso de anónimo: los usuarios que se les asigne este permiso solamente tendrá acceso a la web quedándole totalmente restringido el acceso a la administración.

6.4.5. Administración de Usuarios

Cuando se accede a la sección de Usuarios primero se ve el listado de los Grupos de usuarios que contienen los diferentes usuarios definidos Figura 6.14.

Dando la opción de crear un nuevo grupo o modificar uno existente Figura 6.15, o de acceder al grupo para ver el listado de las Permisos que contiene Figura 6.16. En la edición de los

Grupos de usuarios Figura 6.15 se escogen los permisos que se quieren otorgar a todos los usuarios que contenga dicho grupo.

Figura 6.14: Listado de los Grupos de Usuarios.

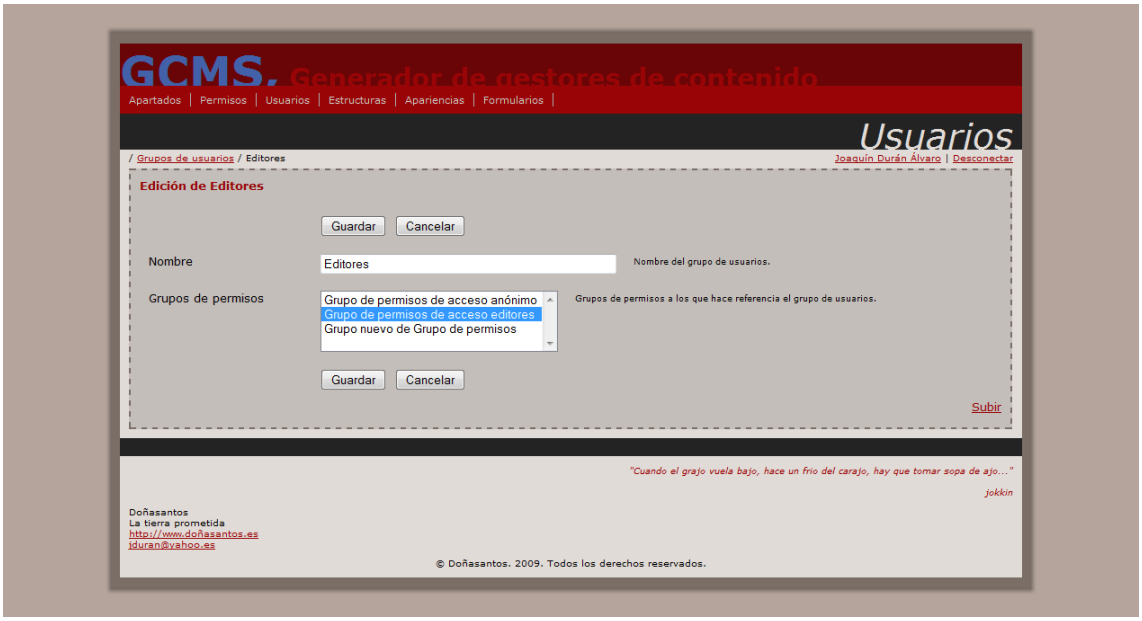


Figura 6.15: Edición de un Grupo de Usuarios.

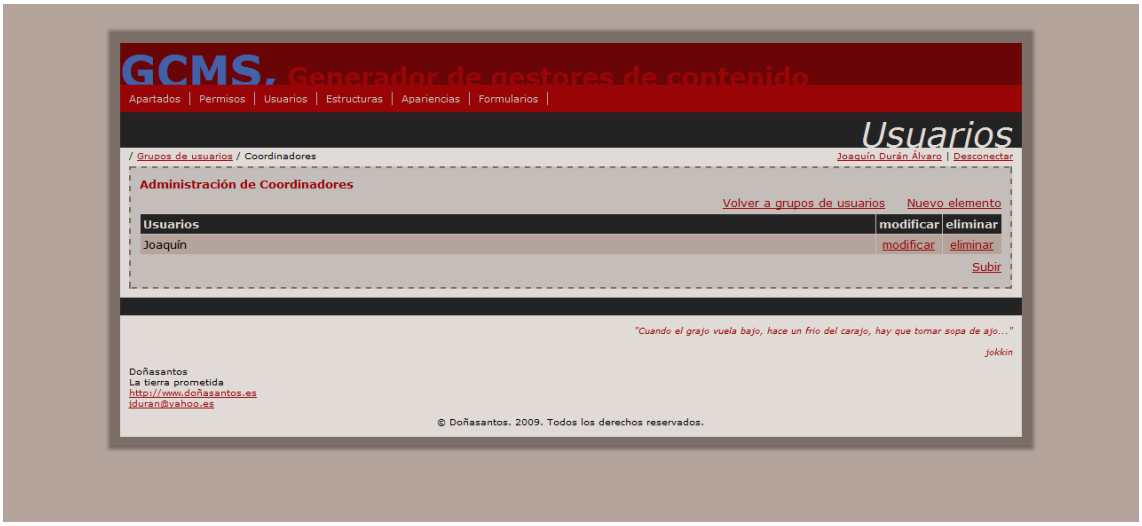


Figura 6.16: Listado de un grupos de Usuarios.

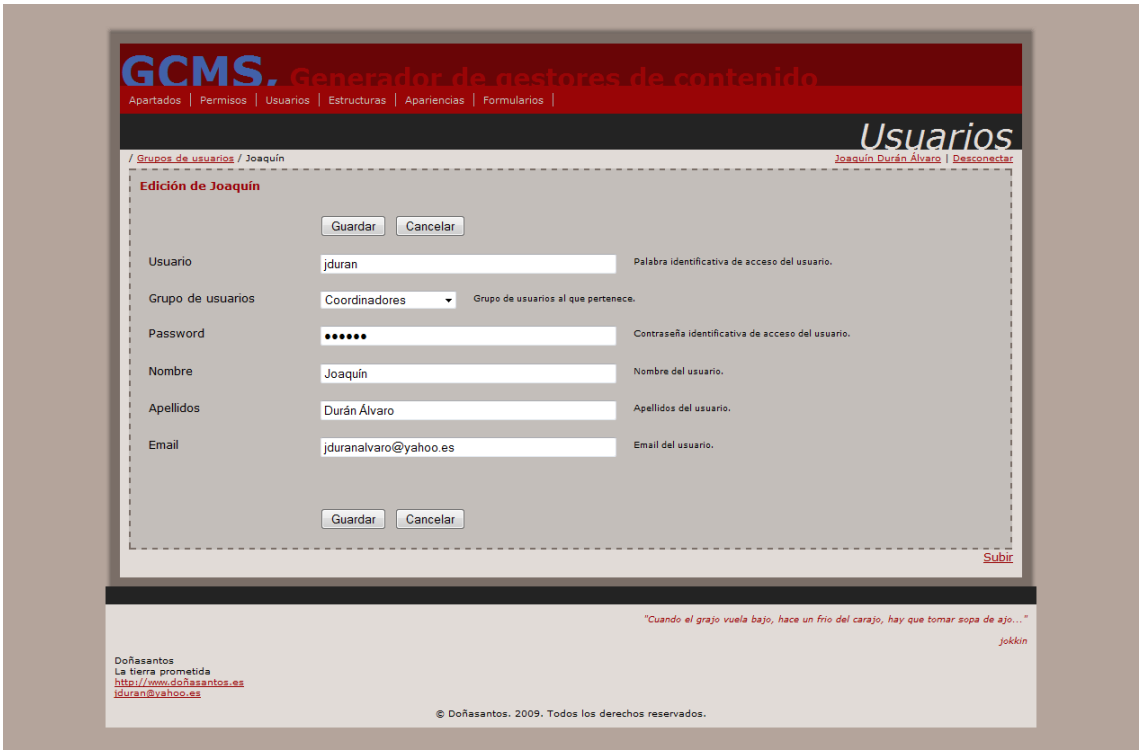


Figura 6.17: Edición de un Usuario.

Los Grupos de Usuarios que se dejan guardados en la implementación son:

- **Administrador:** relacionándolo con el Grupo de Permiso de administración
- **Editor:** relacionándolo con el Grupo de Permiso de editor.
- **Anónimo:** relacionándolo con el Grupo de Permiso de anónimo.

6.4.6. Administración de Apariencias

Cuando se accede a la sección de Apariencias se ve el listado de las Apariencias Figura 6.18. Estas apariencias representan los diseños gráficos que pueden tener la web generada por el CMS. Se pueden crear, modificar y activar Figura 6.19. Cuando se edita una apariencia se puede definir los colores, imágenes, tamaños y tipografías de los textos del contenido gráfico de la web. Para que una apariencia se vea en la web esta tiene que estar activada, pudiendo haber solamente una apariencia activada a la vez, cuando se activa una, directamente se desactivan el resto de Apariencias.

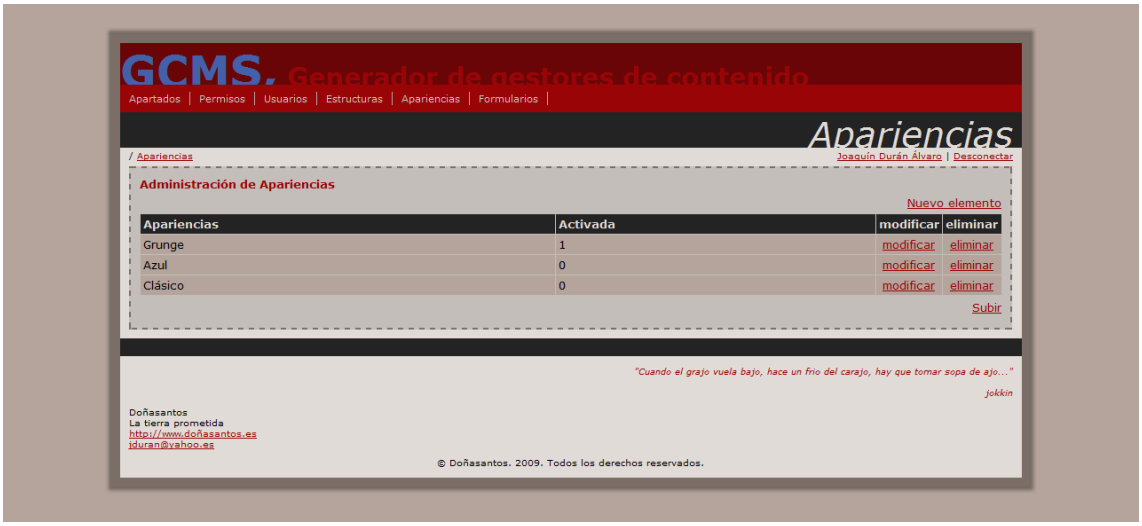
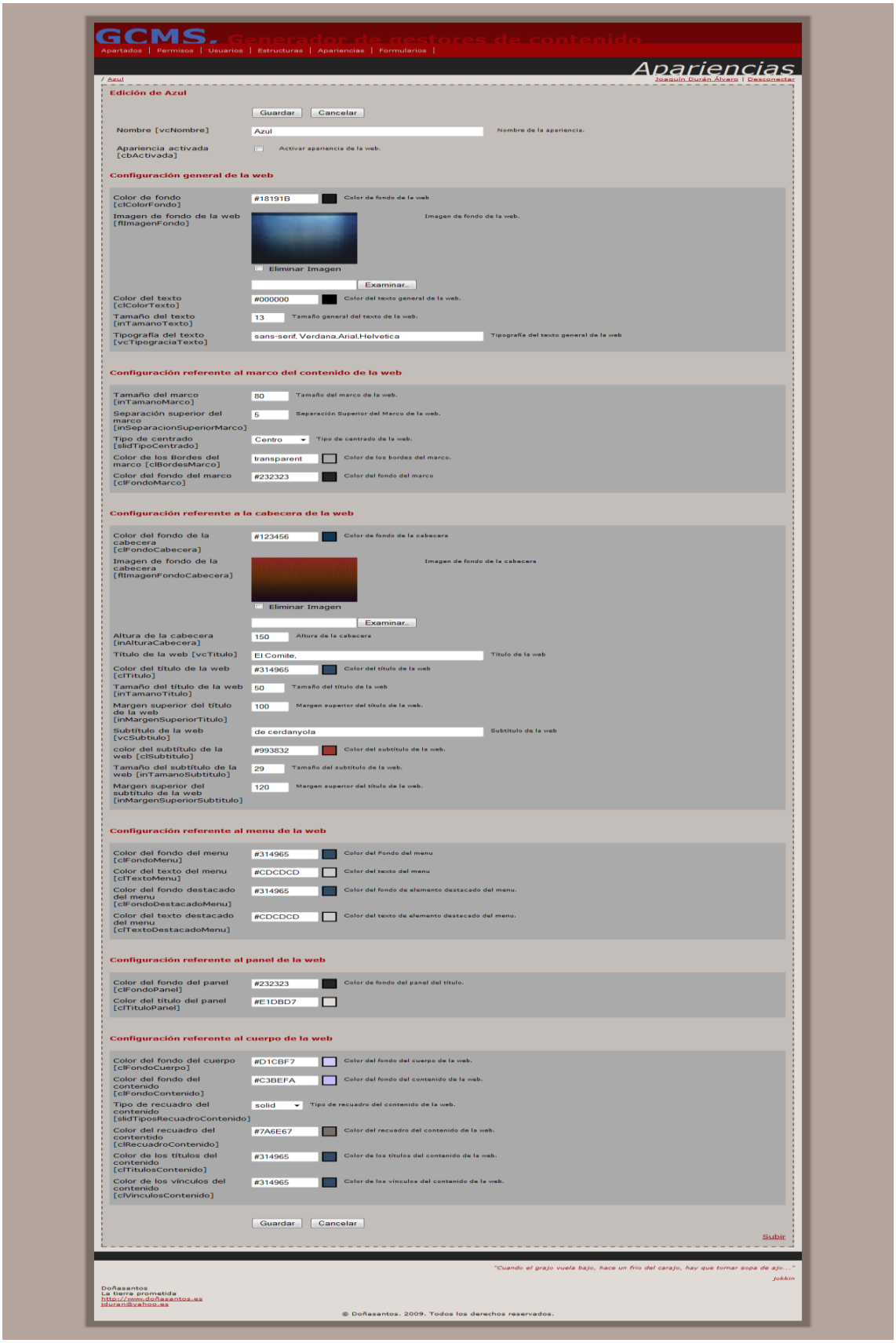


Figura 6.18: Listado de las Apariencias.



Figrua 6.19: Edición de una apariencia.

6.4.7. Administración de Formularios

Al accede a la sección de Formularios se ve el listado de Formularios que contienen las Recolecciones de formularios Figura 6.23. Dando la opción de crear un nuevo Formulario o modificar uno existente Figura 6.21, o de acceder para ver el listado de las Recolecciones del Formulario 6.22.

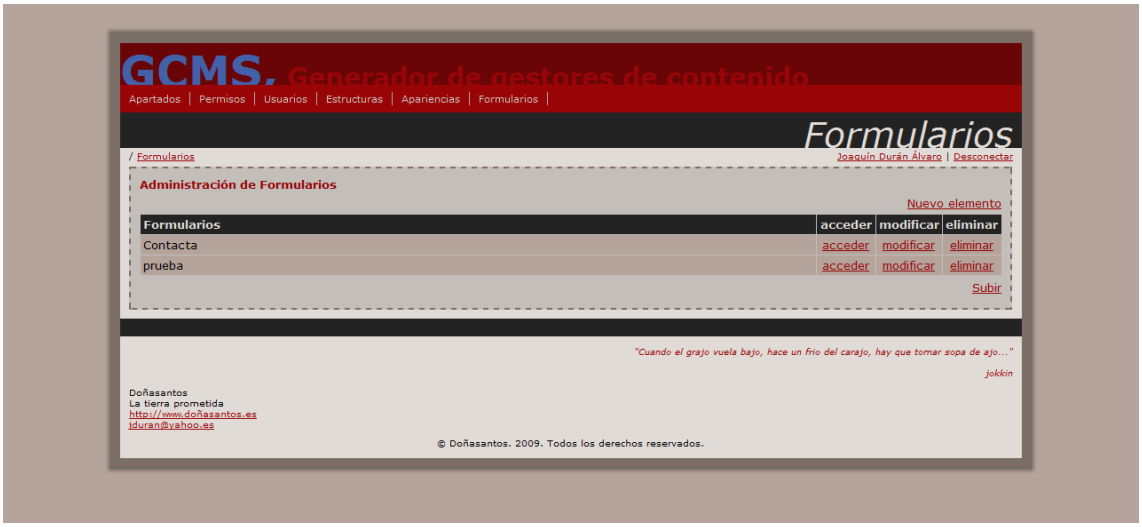


Figura 6.20: Listado de los Formularios.

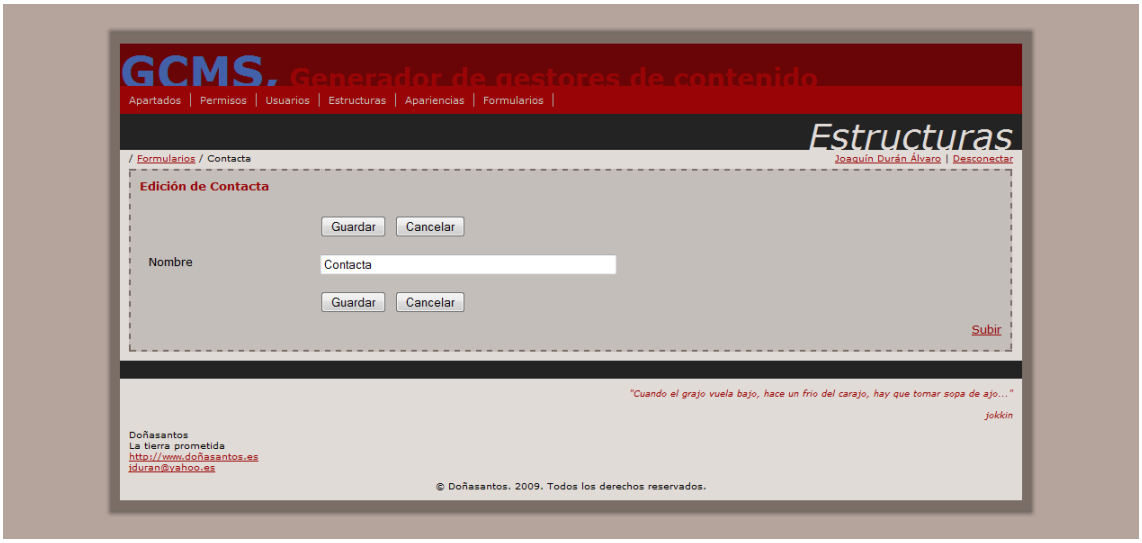


Figura 6.21: Edición de un Formulario.

Una Recolección de Formulario representa los datos que un usuario ha enviado a través del formulario de la web vinculado al elemento Formulario de la administración donde se encuentra agrupada la Recolección. Esta vinculación se realiza al editar un Apartado que contiene un Formulario indicándole a que elemento Formulario de la administración se quiere que lleguen los datos enviados en Forma de Recolección de Formulario.

Figura 6.22: Listado de las Recolecciones de un Formulario.

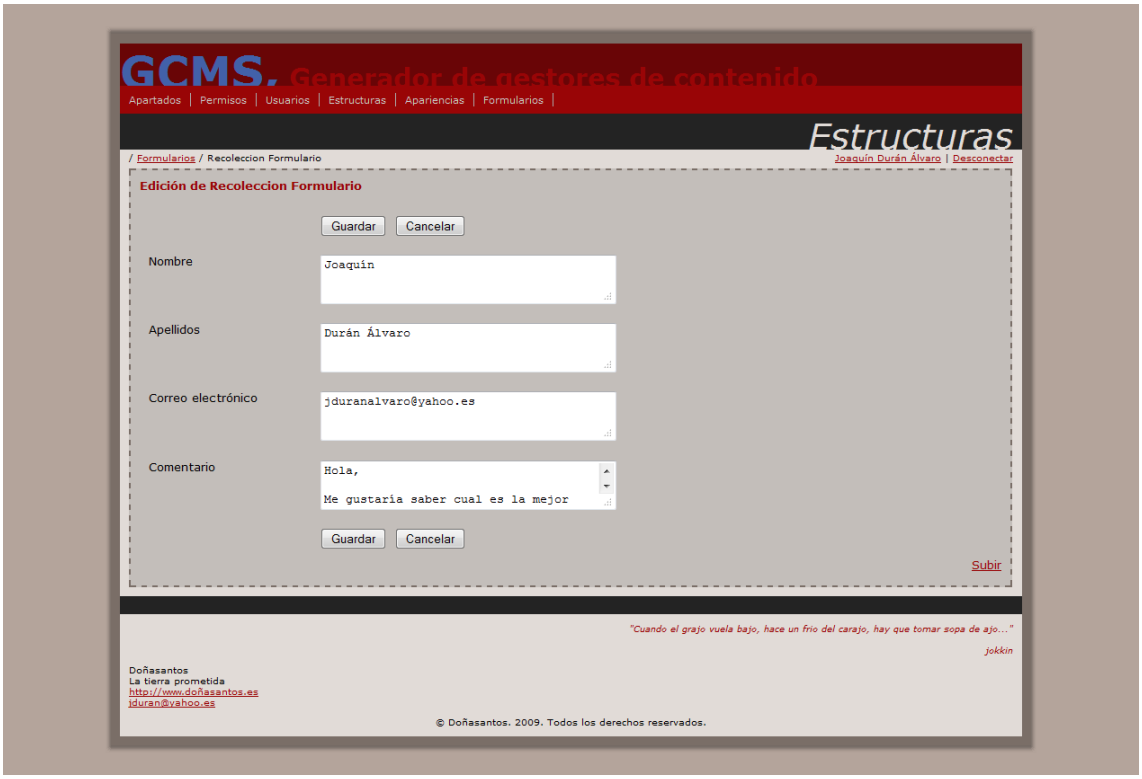


Figura 6.23: Edición de una Recolección de un Formulario.

6.5. Batería de pruebas

A continuación se muestra el test de comprobación de todas las funcionalidades del CMS, indicando la comprobación de que funcionan correctamente. Con este tipo de test se pretende tener controlados todos los posibles errores de la aplicación y evitarlos.

6.5.1. Identificación del usuario

Nº	Funcionalidades	Funcionamiento
1.	Identificación de usuario existente en la base de datos.	OK
2.	Mostrar mensaje de error si se intenta acceder sin introducir el usuario o la contraseña.	OK
3.	Mostrar mensaje de error si se intenta acceder con un usuario o contraseña incorrectos.	OK
4.	Guardar los permisos vinculados al usuario identificado, aplicándole las restricciones que le corresponde mientras navega por la aplicación.	OK

6.5.2. Administración de Estructuras

Nº	Funcionalidades	Funcionamiento
1.	Ver listado de Grupos de estructuras.	OK
2.	Crear un nuevo Grupo de estructuras.	OK
3.	Modificar un Grupo de estructuras.	OK
4.	Eliminar un Grupo de estructuras.	OK
5.	Acceder a un Grupo de estructuras.	OK
6.	Ver listado de Estructuras.	OK
7.	Crear una nueva Estructura.	OK
8.	Modificar una Estructura.	OK
9.	Eliminar una Estructura.	OK
10.	Pre-visualizar una Estructura.	OK
11.	En la edición de una Estructura ver su listado de Componentes.	OK
12.	Añadir un Componente a una Estructura.	OK
13.	Modificar los Componentes de una Estructura.	OK
14.	Eliminar un Componente de una Estructura.	OK

6.5.3. Administración de Apartados

Nº	Funcionalidades	Funcionamiento
1.	Ver listado de Apartados.	OK
2.	Crear un Apartado eligiendo un tipo de Estructura.	OK
3.	Modificar un Apartado.	OK
4.	En la edición de un Apartado que contenga un formulario vincularlo con un Formulario de la administración.	
4.	Eliminar un Apartado.	OK
5.	Pre-visualizar un Apartado.	OK
6.	Acceder dentro de un Apartado.	OK
7.	Ver el listado de Apartados contenidos dentro de un Apartado.	OK

6.5.4. Administración de Permisos

Nº	Funcionalidades	Funcionamiento
1.	Ver listado de Grupos de permisos.	OK
2.	Crear un nuevo Grupo de permisos.	OK
3.	Modificar un Grupo de permisos.	OK
4.	Eliminar un Grupo de permisos.	OK
5.	Acceder a un Grupo de permisos.	OK
6.	Ver listado de Permisos.	OK
7.	Crear un nuevo Permiso.	OK
8.	Modificar un Permiso.	OK
9.	Eliminar un Permiso.	OK

6.5.5. Administración de Usuarios

Nº	Funcionalidades	Funcionamiento
1.	Ver listado de Grupos de usuarios.	OK
2.	Crear un nuevo Grupo de usuarios.	OK
3.	Modificar un Grupo de usuarios.	OK
4.	Vincular un Grupo de usuarios con un Grupo de permisos.	OK
5.	Eliminar un Grupo de usuarios.	OK
6.	Acceder a un Grupo de usuarios.	OK
7.	Ver listado de Usuario.	OK
8.	Crear un nuevo Usuario.	OK
9.	Modificar un Usuario.	OK
10.	Eliminar un Usuario.	OK

6.5.6. Administración de Apariencias

Nº	Funcionalidades	Funcionamiento
1.	Ver listado de Apariencias.	OK
2.	Crear una nueva Apariencia.	OK
3.	Modificar una Apariencia.	OK
4.	Eliminar una Apariencia.	OK
5.	En la edición de una Apariencia activarla.	OK
6.	Al activar una Apariencia que se desactiven el resto.	OK

6.5.7. Administración de Formularios

Nº	Funcionalidades	Funcionamiento
1.	Ver listado de Formularios.	OK
2.	Crear un nuevo Formulario.	OK
3.	Modificar un Formulario.	OK
4.	Eliminar un Formulario.	OK
5.	Acceder a un Formulario.	OK
6.	Ver listado de la Recolección de un Formulario.	OK
7.	Modificar una Recolección de formulario.	OK
8.	Eliminar una Recolección de formulario.	OK

6.5.8. Acceder a la web

Nº	Funcionalidades	Funcionamiento
1.	Ver página de inicio con un listado de resúmenes de la primera capa de Apartados generados por el CMS.	OK
2.	Ver menú con los Apartados generados por el CMS.	OK
3.	Acceder a los diferentes Apartados Generados por el CMS.	OK
4.	Enviar a través de un formulario una Recolección de formulario.	OK
5.	Mostar mensaje de error si se intenta acceder a un Apartado restringido en el que no se tiene permisos para acceder.	OK
6.	Ver correctamente los Apartados generados por el CMS	OK

7. Ejemplos de webs creados

En este capítulo se describen los pasos seguidos para la creación de dos tipos de webs diferentes utilizando la aplicación desarrollada. Estos dos ejemplos son: una web orientada a la promoción de un pueblo de Burgos y el espacio virtual de acceso restringido de un alumno universitario.

7.1. Preparación de los entornos

Para poder empezar a definir las webs que se desean administrar primero se tiene que preparar el entorno del servidor que las alojará, para ello el servidor ha de tener instalados y preparados el servidor tomcat y servidor de bases de datos. A continuación se genera mediante el editor netbeans el fichero del tipo war de la aplicación que se ha programado y se copia en el directorio "webapps" del tomcat, el cual generará automáticamente un directorio con el nombre del fichero del tipo war que será a donde se accederá cuando el usuario entre a la aplicación desde un navegador web. Dentro de la carpeta "webapps" también se ha de crear los directorios donde la aplicación guardará las imágenes y ficheros que se adjunten desde ella, el nombre de estos directorios se configuran en la codificación de la aplicación en los casos de los ejemplos son: "Proyecto/ Donasantos/ Archivos", "Proyecto/ Donasantos/ Ficheros/ apariencias", "Proyecto/ Donasantos/ Ficheros/ apartados", "Proyecto/ Agendauni/ Archivos", "Proyecto/ Agendauni / Ficheros/ apariencias" y "Proyecto/ Agendauni / Ficheros/ apartados". Para finalizar también se realizará una copia de la base de datos en el servidor mysql, tanto de la estructura de la base de datos como de los contenidos, así no se empezará a administrar la web desde cero empezando con un ejemplo de web pudiéndola cambiar a nuestro antojo. A partir de este momento se podrá acceder a las Urls indicadas que apunten a la aplicación para acceder a la web las misma url más la indicació "/admin" para acceder a la zona de administración para poder empezar a definir las webs.

7.2. Web orientada a la promoción de un pueblo de Burgos

Se pretende construir una web que sirva para promocionar un pueblo de Burgos llamado Doña Santos. La web estará compuesta de una serie de secciones relacionadas con el pueblo, típico de una web de turismo: noticias, artículos, fotos, agenda, recetas, formulario de contacto, indicaciones de la localización del pueblo y rutas. Por el momento se efectúa la instalación en el ordenador de pruebas hasta que se defina en que servidor de internet se alojará la web, y para acceder a la web se introduce en la url de un navegador la siguiente dirección: http://localhost:8080/Donasantos ya la zona de administración la siguiente dirección: http://localhost:8080/Donasantos/admin/.

7.2.1. Mapa web

Mapa web el donde se ve las diferentes opciones que tiene la web generada, Figura 7.1.

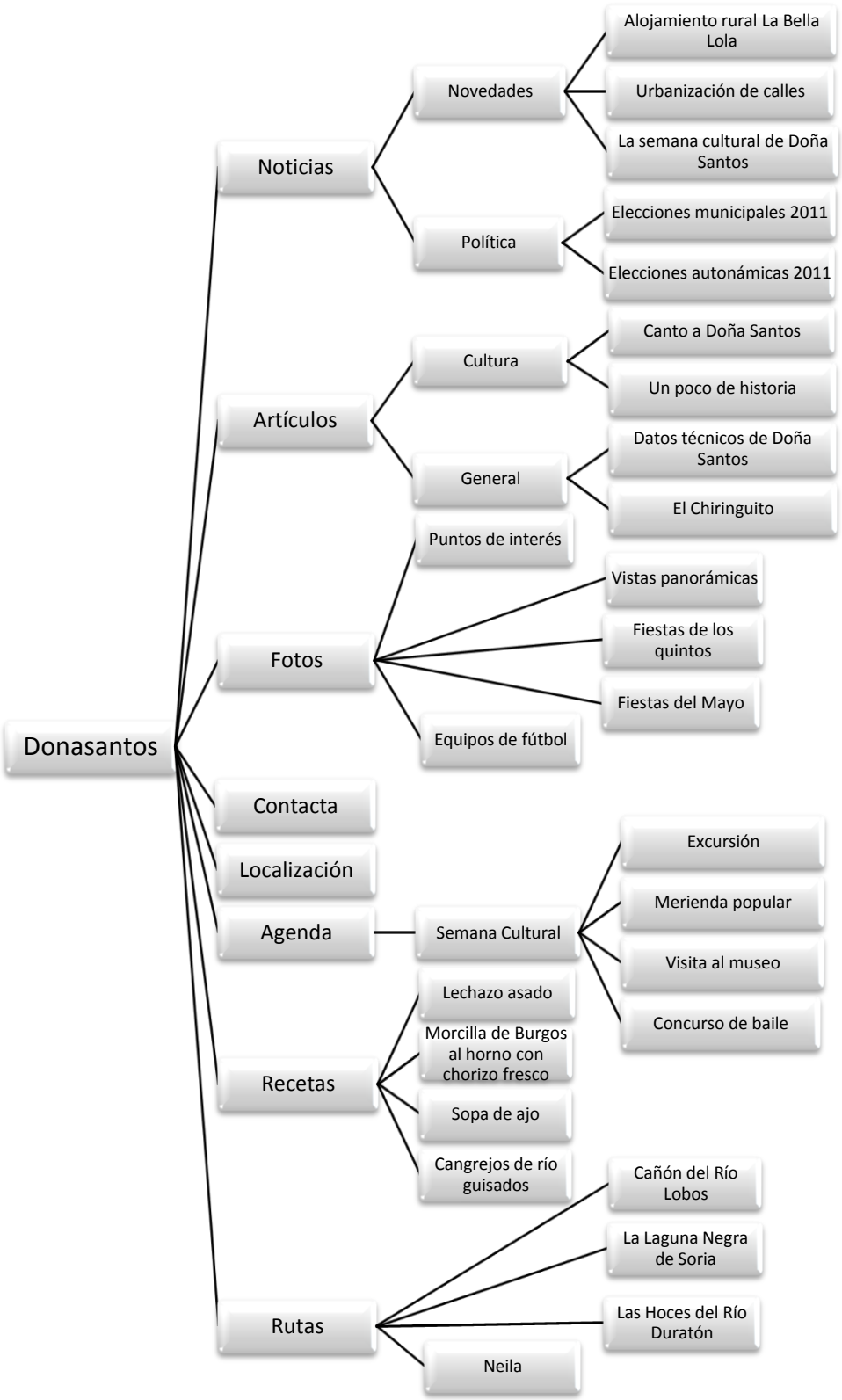


Figura 7.1: Mapa web.

7.2.2. Definición de las estructuras de la web

La web usará las siguientes Estructuras genéricas definidas en la implementación del CMS explicado en el apartado *6.4.2. Administración de Estructuras* del capítulo 6: sección Figura 7.2, noticia Figura 7.3, artículo Figura 7.4, galería fotográfica Figura 7.5, agenda, evento Figura 7.6. y formulario Figura 7.7.

Figura 7.2: Sección.

Figura 7.3: Noticia.

Figura 7.4: Artículo.

Figura 7.5: Galería fotográfica.

Figura 7.6: Galería fotográfica.

Figura 7.7: Formulario.

Además de las Estructuras genéricas definidas en la implementación del CMS, se va a crear un Grupo de Estructuras llamado Estructuras Doña Santos que contendrá las Estructuras nuevas que se crearan para desarrollar apartados de la web no contemplados en la implementación del CMS. Estas estructuras son:

Recetas: pensada para contener recetas típicas de Doña Santos y consta de los componentes título, foto, ingredientes y preparación, Figura 7.8.

Figura 7.8: Recetas.

.

Localización: tipo de estructura pensada para explicar la posición exacta donde se encuentra el pueblo, está formada por los componentes: título, texto y tres imágenes. Figura 7.9.

Figura 7.9: Localización.

Rutas: pensada para contener información de rutas cercanas al pueblo y está compuesta por los componentes nombre, bloque de texto 1, foto 1, bloque de texto 2, foto 2, bloque de texto 3, foto 3, bloque de texto 4, foto 4, bloque de texto 5 y foto 5. Figura 7.10.

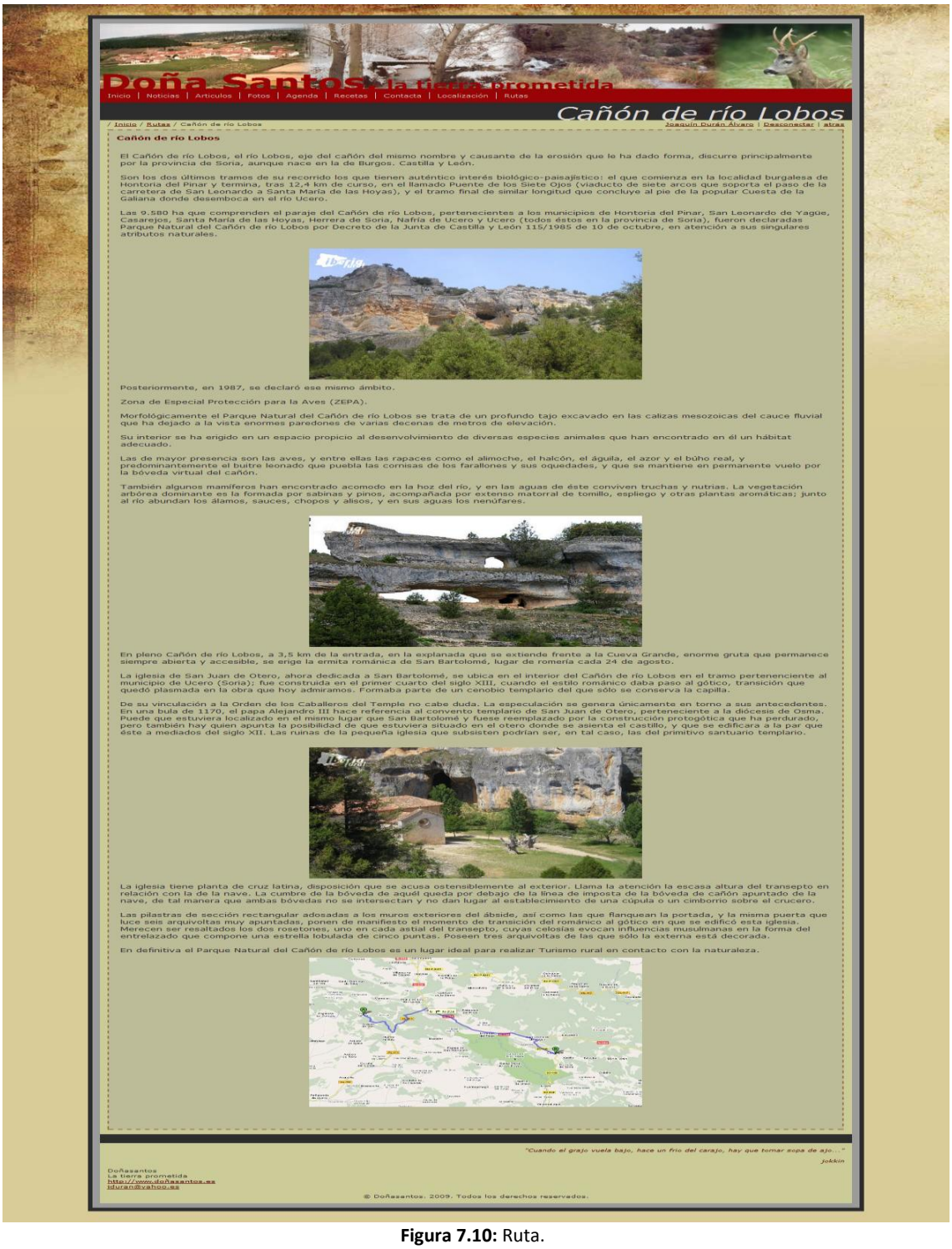


Figura 7.10: Ruta.

7.2.3. Definición de los usuarios y sus permisos

Para definición de los usuarios y permisos el administrador usa los definidos en la implantación del CMS sin realizar ninguna modificación (apartados *6.4.4. Aministración de Permisos* y *5.4.5. Administración de Usuarios* del capítulo 6), estos son los permisos de administración, de editor y de anónimo. Y los usuarios: administrador, editor y anónimo.

7.3. Espacio virtual restringido a un alumno universitario

En este ejemplo de web se pretende construir un espacio virtual de acceso restringido a un alumno universitario donde pueda guardar contenido relacionado con sus estudios. Por el momento se efectúa la instalación en el ordenador de pruebas hasta que se defina en que servidor de internet se alojará la web, y para acceder a la web introduce en la url de un navegador la siguiente dirección: http://localhost:8080/Agendauni ya la zona de administración la siguiente dirección: http://localhost:8080/Agendauni/admin/.

7.3.1. Mapa web

En la Figura 7.11 se muestra un mapa web el donde se ve las diferentes secciones que tiene la web generada.

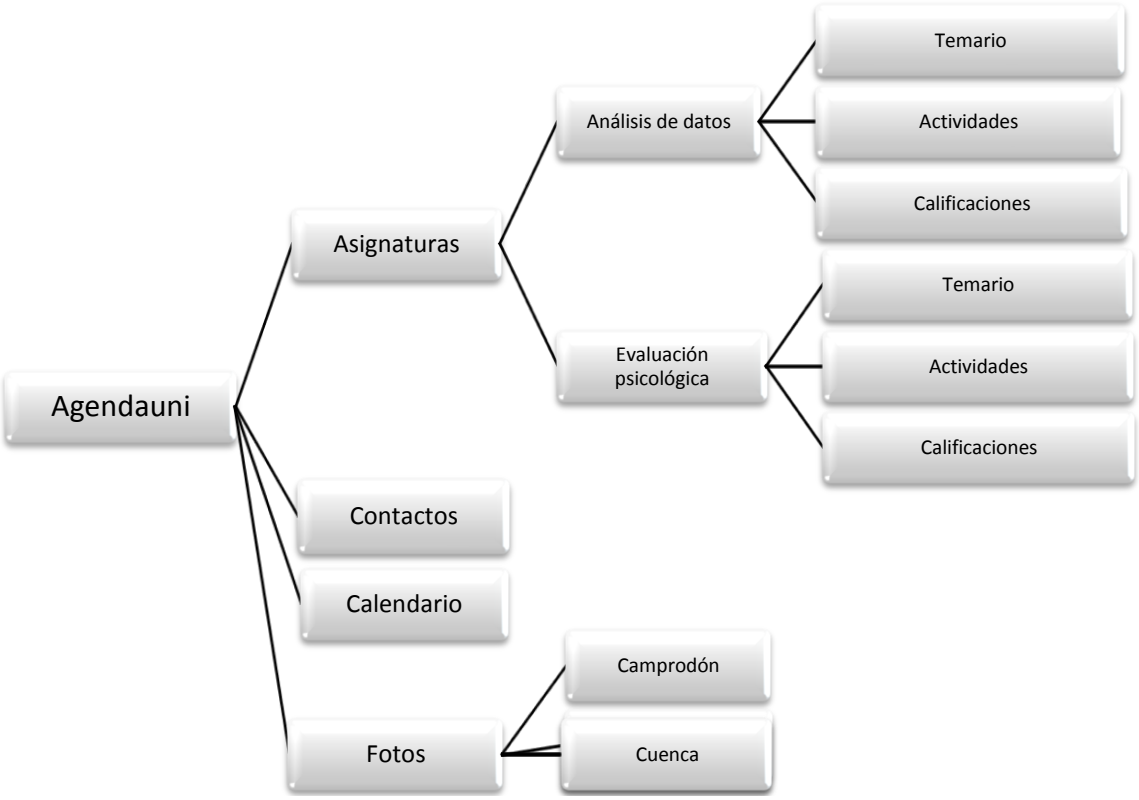


Figura 7.11: Mapa web.

7.3.2. Definición de las estructuras de la web

A diferencia de la web de promoción del pueblo de burgos, está web es de acceso restringido y se cambiarán los permisos del usuario anónimo para que no pueda acceder a los apartados que sean de tipo de Estructura genérica de la web. Creando nuevas Estructuras de carácter público a las que sí que podrá acceder. De esta forma se consigue restringir la mayor parte de la web dejando alguna zona que sea pública. La web aprovecha las siguientes Estructuras genéricas definidas en la implementación del CMS explicado en el capítulo 6: sección Figura 7.12, agenda, eventos Figura 7.14, galería fotográfica Figura 7.15 y galería de archivos Figura 7.16. Las estructuras de carácter público se crean dentro del Grupo de Estructuras llamado Agendauni que contendrá las Estructuras nuevas: sección pública y artículo público Figura 7.13.

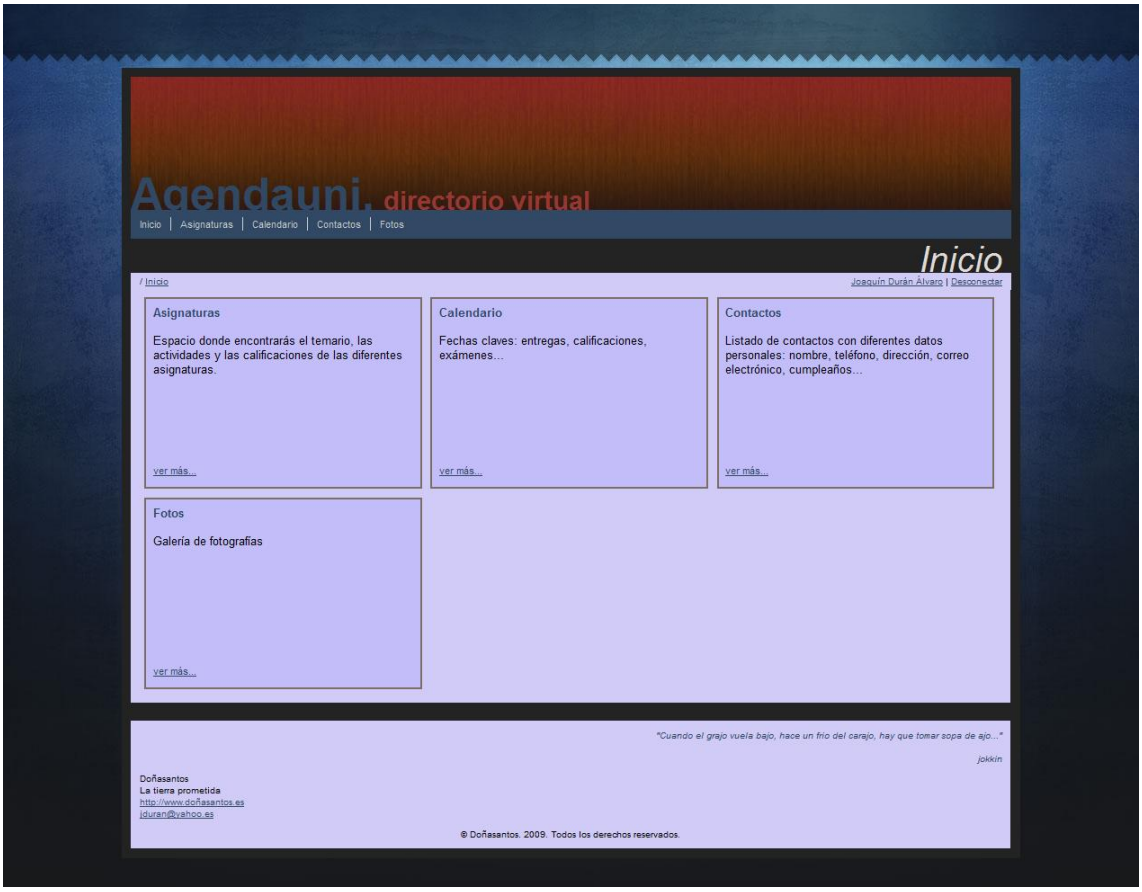


Figura 7.12: Página de inicio mostrando las secciones de acceso restringido.

Figura 7.13 Página de inicio mostrando las secciones de acceso público.

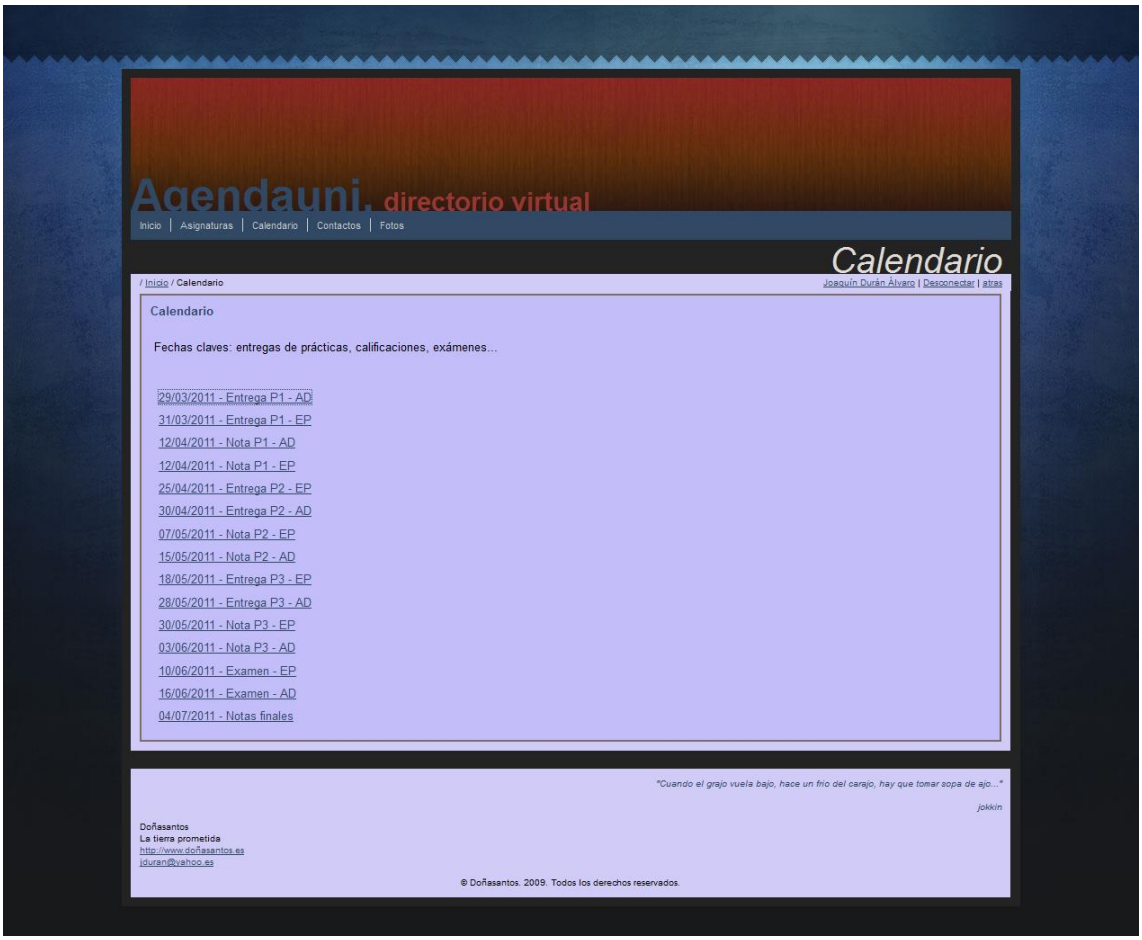


Figura 7.14: Calendario.

Figura 7.15: Galería fotográfica.

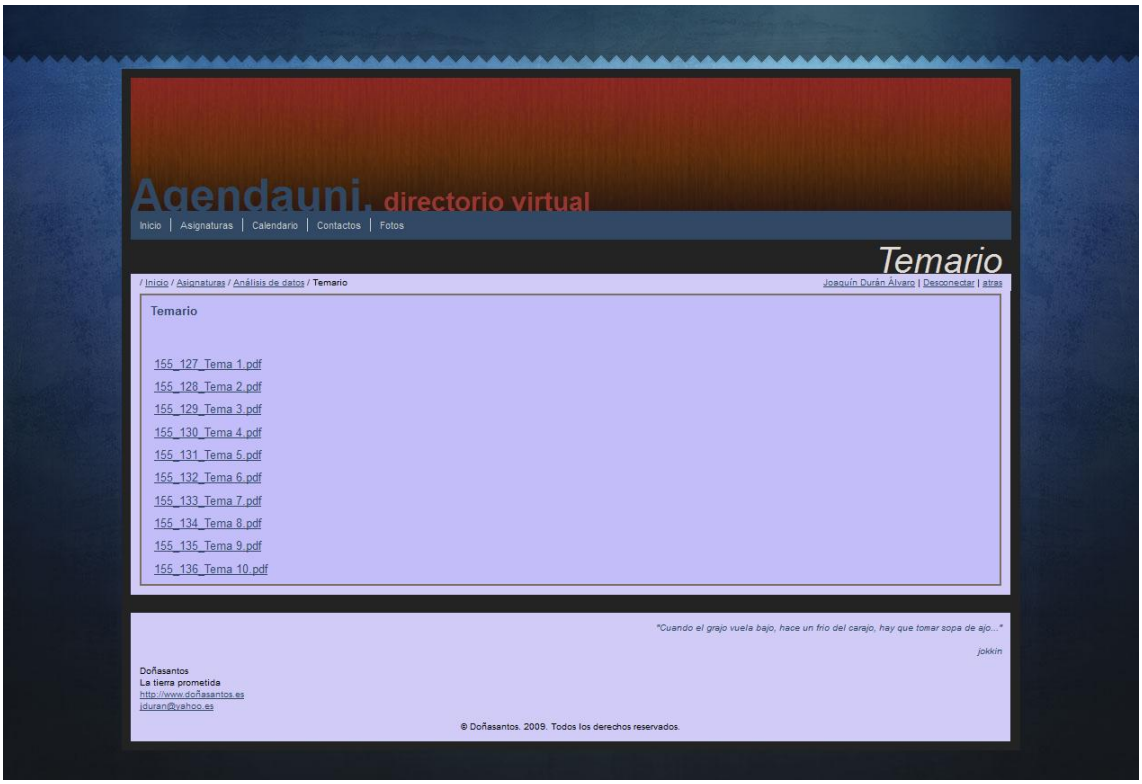


Figura 7.16: Galería de archivos.

7.3.3. Definición de usuarios y sus permisos

Para definición de los usuarios y permisos el administrador usa los definidos en la implantación del CMS (apartados *6.4.4. Administración de Permisos* y *5.4.5. Administración de Usuarios* del capítulo 6), modificando los permisos de anónimo para que solamente pueda acceder y visualizar los apartados que utilizan las nuevas estructuras sección pública y articulo público. Otro cambio que se realiza es la modificación del nombre del Grupo de Usuarios Editor por Alumno.

8. Conclusiones

En la realización del proyecto además de desarrollar una herramienta que se podré utilizar en mi desempeño laboral y que irá creciendo cada vez más, se ha conseguido dar un paso más dentro del mundo de la programación web aprendiendo a trabajar utilizando la metodología de los frameworks, esto se considera importante porque que una de las finalidades de estas herramientas es la de recopilar lo que se piensa que son buenos hábitos de trabajo a la hora de diseñar, desarrollar e implementar un proyecto y conseguir que los desarrolladores los apliquen.

Como objetivos cumplidos se mencionan los siguientes aspectos:

- Obtención de los conocimientos sobre el framework Struts y su metodología de trabajo MVC.
- Realización de un CMS que ayude a crear páginas web sin tener que programarlas.
- Que al desarrollador le resulte fácil programar e incorporar nuevas funcionalidades en el CMS.
- Crear dos ejemplos de webs diferentes a través de la aplicación desarrollada.
- Diseño e implementación de una base de datos que permita gestionar la información de la aplicación.
- Ofrecer un sistema de seguridad robusto que solamente permita acceder a las áreas restringidas a los usuarios que tengan permiso para ello.
- Realización de una aplicación que no sobrecargue los recursos del hardware sobre el que se ejecutará.

Como objetivos no cumplidos se pueden mencionar:

- Para la realización del CMS se escogió el lenguaje Java sobre el lenguaje PHP porque ofrece la posibilidad de realizar funcionalidades más potentes, como por ejemplo la opción de poder instalar la web en el servidor desde un programa de escritorio desde nuestro ordenador o la realización de un chat en la web. Cosa que al final no se ha aprovechado ya que todas las funcionalidades que actualmente ofrece el CMS se pueden programar con PHP.
- Se piensa que se puede mejorar la interfaz gráfica de la aplicación consiguiendo que sea más práctica y fácil de utilizar, incorporando funcionalidades como menús dinámicos y efectos gráficos.
- No se ha desplegado la aplicación en un entorno de producción adecuado para su utilización, estando actualmente únicamente en un entorno de desarrollo.
- No se ha conseguido desarrollar el proyecto dentro del plazo de tiempo establecido inicialmente, teniendo que invertir muchas más horas de las calculadas inicialmente.

8.1. Planificación real de la realización del proyecto

En la realización del proyecto se ha sufrido un retraso respecto a la previsión inicial. Sobre todo a la hora de adquirir los conocimientos sobre el framework Struts y a la hora de programar el CMS utilizando dicho framework. Esto se debe a la inexperiencia del desarrollador sobre la programación java. Ya que además de ser un lenguaje diferente al que se suele utilizar, y utilizar el nuevo entorno de desarrollo del que se ha obtenido conocimientos, también cambia la forma de pensar a la hora de diseñar y programar utilizando metodologías como la programación orientada a objetos y la arquitectura MVC. Otro motivo importante que ha influido en el retraso de la realización del proyecto, ha sido la poca disposición de tiempo que se tenía debido a responsabilidades laborales. A continuación en la Figura 8.1 se detalla la plantificación real de la realización del proyecto.

Id	Nombre de tarea	Duración	Comienzo	Fin	Predecesoras	Iniciales del recurso
1	Inicio del proyecto: asignación y matriculación del proyecto	5 horas	lun 16/11/09	mié 18/11/09		DP;JP
2	**Planificación.**	**5 días**	**mié 18/11/09**	**mar 01/12/09**		
3	Estudio de viabilidad	10 horas	mié 18/11/09	mar 01/12/09	1	JP
4	**Investigación de los fundamentos teóricos**	**70 días**	**mar 01/12/09**	**mié 07/04/10**		
5	Investigación de entornos de programación web	20 horas	mar 01/12/09	vie 18/12/09	3	A;P
6	Investigación de Struts	90 horas	vie 18/12/09	vie 12/03/10	5	A;P
7	Investigación de entornos de servidor Linux	10 horas	vie 12/03/10	vie 19/03/10	6	TS
8	Redacción de la documentación de fundamentos teóricos	20 horas	vie 19/03/10	mié 07/04/10	7	TS
9	**Análisis de la aplicación**	**22,5 días**	**mié 07/04/10**	**vie 07/05/10**		
10	Análisis de requisitos (diagrama de casos de uso)	8 horas	mié 07/04/10	mar 13/04/10	8	A
11	Análisis de datos (diagrama de clases)	12 horas	mar 13/04/10	mié 21/04/10	10	A
12	Análisis de seguridad	10 horas	mié 21/04/10	mié 28/04/10	11	A
13	Documentación del análisis	10 horas	mié 28/04/10	mié 05/05/10	12	A
14	Aprobación del análisis (Punt de control).	5 horas	mié 05/05/10	vie 07/05/10	13	A;JP;DP
15	**Diseño de la aplicación**	**28 días**	**lun 10/05/10**	**mié 16/06/10**		
16	Diseño de la base de datos	15 horas	lun 10/05/10	mié 19/05/10	14	A;P
17	Diseño funcionalidad (diagrama de secuencia)	12 horas	mié 19/05/10	jue 27/05/10	16	A;P
18	Diseño de la interfaz (esquemas de las pantallas)	6 horas	jue 27/05/10	mar 01/06/10	17	A;P
19	Diseño de la batería de pruebas (test)	6 horas	mar 01/06/10	vie 04/06/10	18	A;P;TP
20	Documentación del diseño.	12 horas	vie 04/06/10	lun 14/06/10	19	A
21	Aprobación del diseño (Punto de control).	5 horas	lun 14/06/10	mié 16/06/10	20	DP;A;JP
22	**Desarrollo de la aplicación.**	**122 días**	**jue 17/06/10**	**mié 13/04/11**		
23	Preparación del entorno de desarrollo.	12 horas	jue 17/06/10	jue 24/06/10	21	P
24	Configuración de la base de datos.	12 horas	vie 25/06/10	jue 14/10/10	23	P
25	Programación de la aplicación	200 horas	vie 15/10/10	mié 30/03/11	24	P
26	Desarrollo de la interfaz de usuarios.	20 horas	jue 31/03/11	mié 13/04/11	25	P
27	**Test y pruebas.**	**13,5 días**	**jue 14/04/11**	**mar 10/05/11**		
28	Realización de la batería de pruebas	10 horas	jue 14/04/11	mié 27/04/11	26	TP
29	Documentación de desarrollo y test.	12 horas	jue 28/04/11	jue 05/05/11	28	TP
30	Aprobación del desarrollo i pruebas (Punto de control).	5 horas	vie 06/05/11	mar 10/05/11	29	JP;A;P;DP
31	**Implantación.**	**17,5 días**	**mar 10/05/11**	**jue 02/06/11**		
32	Utilización de la aplicación para generar webs de ejemplo	20 horas	mar 10/05/11	mar 24/05/11	30	A;P
33	Baterias de pruebas sobre las web de ejemplo	10 horas	mar 24/05/11	mar 31/05/11	32	TP
34	Formación de usuarios.	5 horas	mar 31/05/11	jue 02/06/11	33	A
35	Generación de documentos (memoria del proyecto).	30 horas	vie 03/06/11	mar 28/06/11	34	JP
36	Preparación de la defensa del proyecto	10 horas	mié 29/06/11	mar 05/07/11	35	DP;JP

Figura 8.1: plantificación real de la realización del proyecto.

Se acompaña la planificación con el diagrama de gantt Figura 8.2 y 8.3 que muestra de manera gráfica el tiempo dedicado para las diferentes tareas.

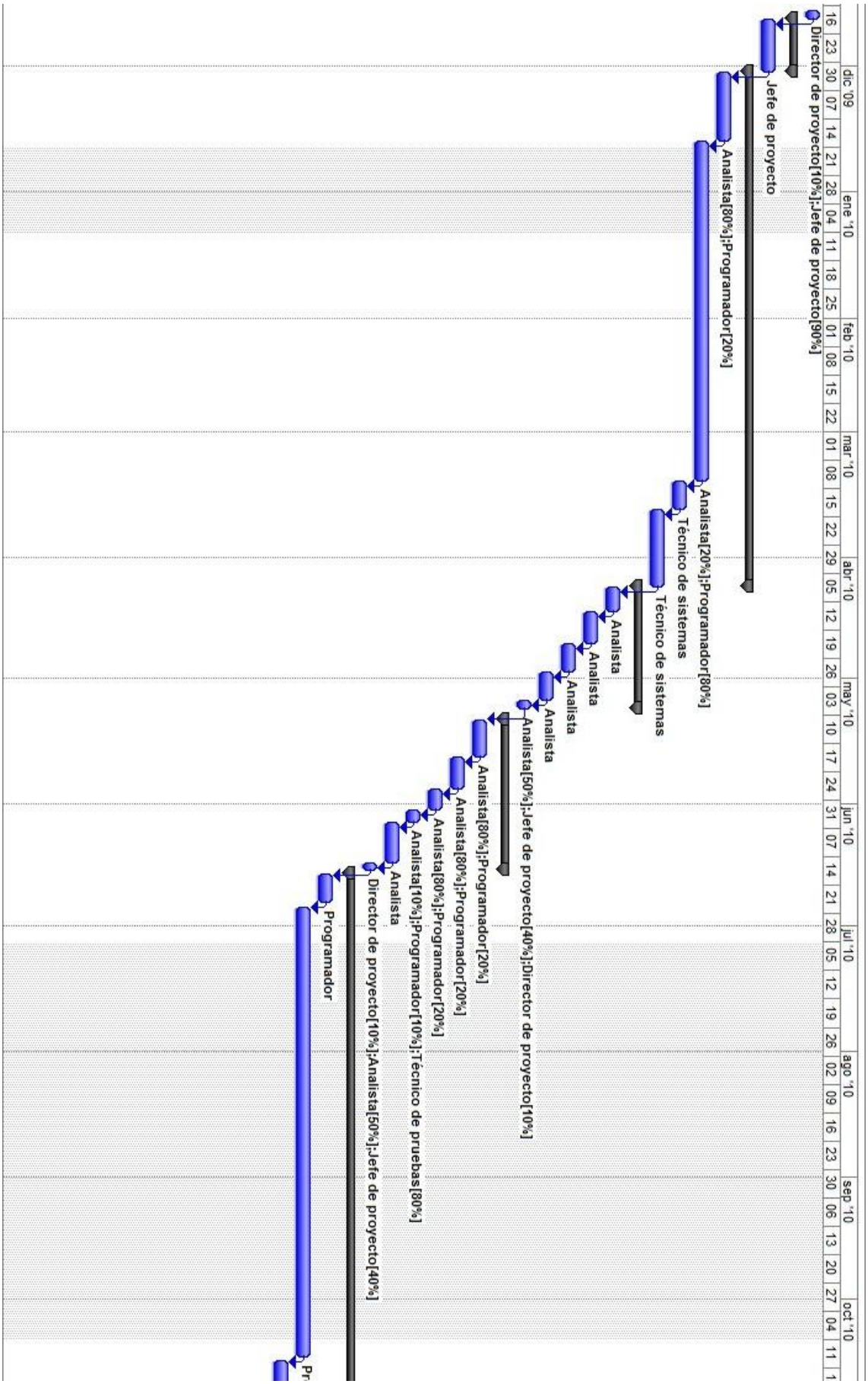


Figura 8.2: Diagrama de Gantt.

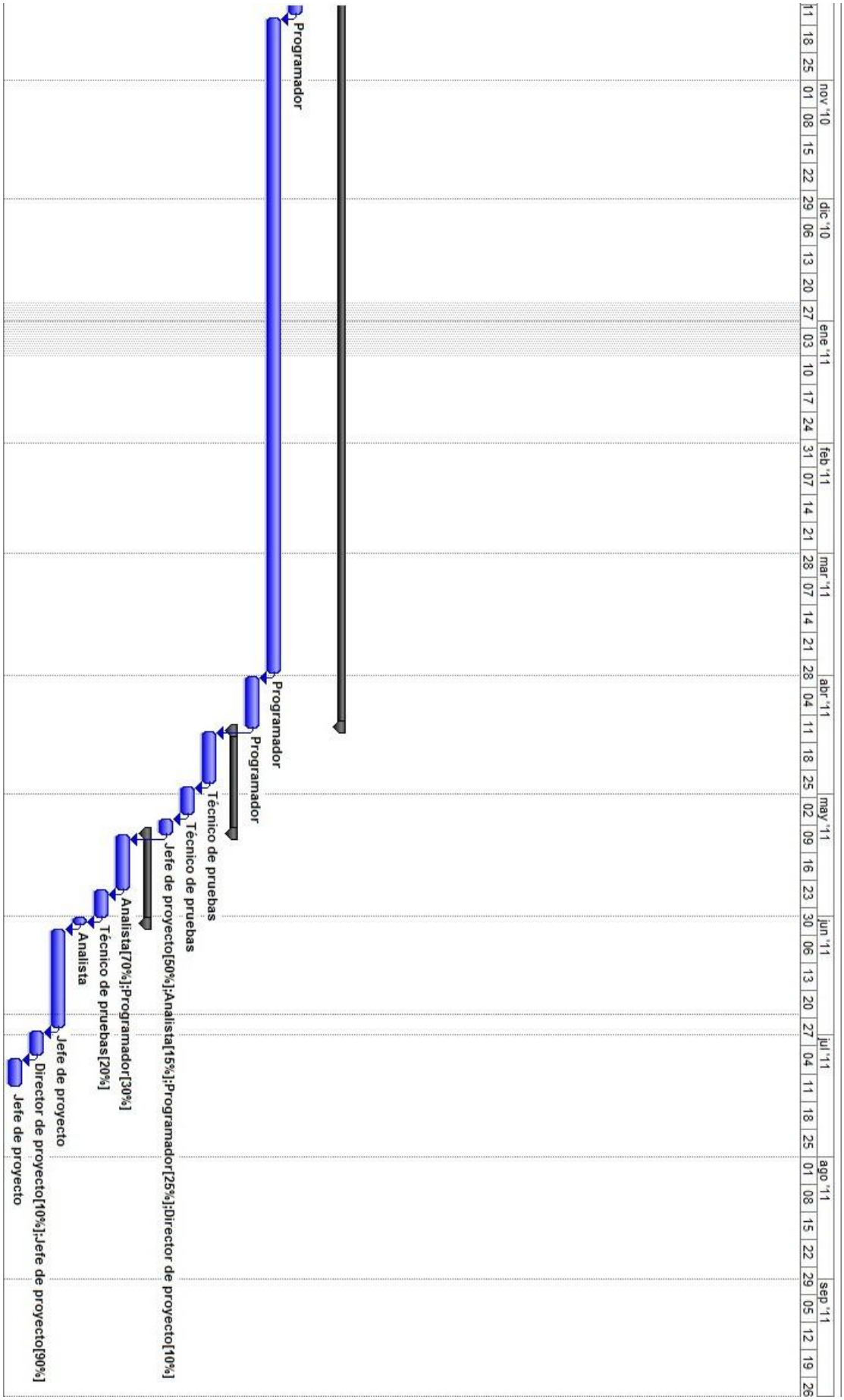


Figura 8.2: Diagrama de Gantt.

8.2. Posibles mejoras o ampliaciones

Desde un principio se ha realizado la aplicación con la idea de que siempre tenga la opción de crecer, pero como actualizaciones y mejoras más inmediatas se contemplan las siguientes opciones:

- Añadir más utilidades de la aplicación que mejoren la administración de la web como poder ordenar los listados con diferentes criterios, copiar elementos, mover elementos de un sitio a otro, que un elemento pueda tener más de una ubicación a la vez, poder gestionar las columnas que se quiere que aparezcan en los listados de la administración.
  Por ejemplo si se quiere que una noticia esté en más de una sección, que no haga falta editarla dos veces, si no que se dé la opción de copiarla o simplemente indicar que se encuentra en las dos secciones a la vez, así si se quiere modificar, con una vez que se edite será suficiente.
- Mejorar el entorno visual de la aplicación con menús dinámicos y efectos visuales que le den un aspecto más profesional. Como por ejemplo incorporar un menú en la sección de apartados con formato de árbol emulando un sistema de carpetas y mostrando más de una capa profundidad, consiguiendo poder acceder directamente a un apartado sin tener que pasar por todos sus predecesores.
- Programar un entorno de instalación que se pueda ejecutar desde el escritorio de un ordenador personal y permita instalar la aplicación y la base de datos en un servidor.
- Programar más funcionalidades para las webs como por ejemplo un chat.

Bibliografía

Bibliografía Física

Programador Certificado Java 2 "Curso práctico" (2ª Edición)
De Antonio J. Martín Sierra, (Editorial Ra-MA) 2007.
Iniciación a la programación Java fácil de entender y de seguir, nivel básico medio.

Struts
De Antonio J.Martín Sierra, (Editorial Ra-MA) 2008
Guía práctica con gran variedad ejemplos de programación, orientado al framework en java llamado struts, nivel básico medio.

Sun Certified Programmer for Java 6 Study Guide
De Kathy Sierra y Bert Bates, (Editorial Mc Graw Hill)
Libro oficial de la programación certificada de Sun para Java, libro muy completo, nivel medio alto.

Unix y Linux "Guía práctica" (3ª Edición)
De Sebastián Sánchez Prieto y Óscar García Población, (Editorial Ra-MA) 2007
Manual de documentación sobre los sistemas operativos linux y unix, facil de leer y de comprender, nivel básico medio.

El libro oficial de Ubuntu Server
De Kyle Rankin y Benjamin Mako Hill (Editorial Anaya Multimedia)
Manual de documentación sobre el sistema operativo utilizado como servidor Ubuntu.

Bibliografía On-line

Struts
Disponible: http://struts.apache.org
Último acceso: 18/06/2011

JavaHispano
Disponible en: http://www.javahispano.org/
Ultimo acceso: 18/06/2011
© Copyright 2009, Red Hat Middleware
Portal hispano dedicado al lenguaje de programación java, con numerosa documentación, cometarios y ejemplos.

Spring Source
Disponible en: http://www.springsource.org/
Ultimo acceso: 18/06/2011
Copyright 2009,www.springsource.org
Web oficial del framework en java llamado Spring, con secciones de download, forums y documentación.

Hibernate
Disponible en: https://www.hibernate.org/
Ultimo acceso: 18/06/2011
© Copyright 2009, Red Hat Middleware
Web oficial del framework en java llamado Hibernate, con secciones de download, forums y documentación.

CMS
Disponible en: http://es.wikipedia.org
Ultimo acceso: 01/06/2011
Explicación de conceptos y definición de un CMS.