



# UNIVERSIDADE DA CORUÑA

FACULTADE DE INFORMÁTICA

*Departamento de Computación*

Proxecto de Fin de Carreira de Enxeñería Informática

“Desarrollo de un sistema de información geográfica participativo basado en  
AJAX y el estándar Web Map Service (WMS)”

AUTOR: Jesús Ángel Pérez-Roca Fernández  
TUTOR: Miguel Ángel Rodríguez Luaces

A Coruña, Enero de 2008



## **ESPECIFICACIÓN**

TÍTULO:      *“Desarrollo de un sistema de información geográfica participativo basado en AJAX y el estándar Web Map Service (WMS)”*

CLASE:      *Clásico de Ingeniería*

AUTOR:      *Jesús Ángel Pérez-Roca Fernández*

TUTOR:      *Miguel Ángel Rodríguez Luaces*

TRIBUNAL:

FECHA DE

LECTURA:

CALIFICACIÓN:



## **AUTORIZACIÓN DE ENTREGA**

D. Miguel Ángel Rodríguez Luaces

## **CERTIFICA**

Que el proyecto titulado “Desarrollo de un sistema de información geográfica participativo basado en AJAX y el estándar Web Map Service (WMS)” ha sido realizado por Jesús Ángel Pérez-Roca Fernández, con D.N.I. 53.301.968-C, bajo la dirección de D. Miguel Ángel Rodríguez Luaces. Esta memoria constituye la documentación que, con mi autorización, entrega dicho alumno para optar a la titulación de Ingeniero en Informática.

A Coruña, Enero de 2008

Firmado: D. Miguel Ángel Rodríguez Luaces



## **AGRADECIMIENTOS**

En primer lugar, me gustaría mencionar a mi familia, que siempre ha estado a mi lado cuando lo he necesitado y sin la que nada de esto habría sido posible.

También me gustaría agradecerle a mi tutor de proyecto, Miguel Ángel Rodríguez Luaces por el tiempo y dedicación que me prestó durante el desarrollo de este proyecto, así como por su paciencia y comprensión.

Por último, pero no por ello menos importante, me gustaría agradecer a todas las personas, tanto alumnos, como profesores, como demás personal de la Universidad que han formado parte de esta maravillosa etapa de mi vida que está a punto de terminar para dejar paso a otra etapa distinta como es la vida laboral.



## **RESUMEN**

El objetivo principal de este proyecto es el análisis, diseño e implementación de una aplicación Web para la consulta y visualización de información geográfica proveniente de un servicio Web que implemente el estándar WMS.

Una característica importante de la aplicación es su carácter participativo. La aplicación no se limitará a visualizar información, sino que usuarios registrados en el sistema podrán añadir nuevos elementos de información libremente. Además, la aplicación se diseñará utilizando la tecnología AJAX para permitir un alto grado de interactividad imposible de alcanzar con aplicaciones web tradicionales.

Esta aplicación deberá ser genérica para lo cual se hará independiente de dominios de aplicación específicos. Además, esta aplicación será fácilmente configurable, y se implementará una herramienta que facilite su administración y configuración.



## PALABRAS CLAVE

- Sistemas de Información Geográfica (SIG)
- Web Map Service (WMS)
- AJAX
- Aplicación Web
- J2EE
- EJB3
- Model-View-Controller (MVC)
- Jakarta Struts
- PostgreSQL



## Índice

<b>1. Introducción</b>	<b>1</b>
1.1. Motivación . . . . .	1
1.2. Objetivos del proyecto . . . . .	2
1.3. Estructura de la memoria . . . . .	3
<b>2. Estándares y tecnologías utilizadas</b>	<b>5</b>
2.1. Web Map Service . . . . .	5
2.2. J2EE . . . . .	6
2.3. EJB3 . . . . .	7
2.4. Struts . . . . .	9
2.5. XML . . . . .	11
2.6. CSS (Cascade Style Sheet) . . . . .	11
2.7. JSP (Java Server Pages) . . . . .	12
2.8. AJAX . . . . .	13
<b>3. Análisis</b>	<b>17</b>

---

<b>4. Diseño e implementación</b>	<b>19</b>
4.1. Arquitectura global . . . . .	20
4.2. Paquete EJBConfiguration . . . . .	20
4.3. Paquete HTMLView . . . . .	21
4.3.1. HTML . . . . .	21
4.3.2. WEB-INF . . . . .	22
4.4. Paquete http.controller . . . . .	22
4.4.1. Paquete actions . . . . .	22
4.4.2. Paquete frontcontroller . . . . .	28
4.4.3. Paquete session . . . . .	29
4.4.4. Paquete util . . . . .	29
4.5. Paquete http.util . . . . .	29
4.6. Paquete http.view.messages . . . . .	29
4.7. Paquete model . . . . .	31
4.7.1. Transfer Object . . . . .	33
4.7.2. Entities . . . . .	33
4.7.3. UserFacade . . . . .	37
4.7.4. SearchFacade . . . . .	38
<b>5. Patrones utilizados</b>	<b>45</b>
5.1. Model View Controller . . . . .	45
5.2. Layers . . . . .	46
5.3. Tiles . . . . .	46
5.4. Transfer Object . . . . .	47
5.5. Business Delegate . . . . .	47

---

5.6.	Page-by-Page Iterator . . . . .	48
5.7.	Service Locator . . . . .	48
5.8.	Facade . . . . .	49
5.9.	Factory method . . . . .	49
5.10.	Composite . . . . .	50
5.11.	Singleton . . . . .	50
5.12.	Chain of Responsability . . . . .	50
5.13.	Front Controller . . . . .	51
<b>6.</b>	<b>Manual de usuario</b>	<b>53</b>
<b>7.</b>	<b>Conclusiones y trabajo futuro</b>	<b>101</b>
<b>A.</b>	<b>Manual de instalación</b>	<b>103</b>
A.1.	Instalación de Java Runtime Environment . . . . .	104
A.2.	Instalación de PostgreSQL . . . . .	104
A.3.	Instalación de ant . . . . .	104
A.4.	Instalación de Struts, JSTL y JDOM . . . . .	105
A.5.	Instalación de JBoss . . . . .	105
A.6.	Instalación de la aplicación . . . . .	105
<b>B.</b>	<b>Contenido del CD adjunto</b>	<b>107</b>
<b>C.</b>	<b>Uso de código licenciado</b>	<b>109</b>
C.1.	J2EE Examples . . . . .	110
C.2.	Javascript Image Cropper . . . . .	110
C.3.	WMSMap . . . . .	111

---

C.4. TinyMCE	111
<b>D. Glosario de acrónimos</b>	<b>113</b>
<b>E. Glosario de términos</b>	<b>115</b>
<b>Bibliografía</b>	<b>119</b>

## Índice de figuras

2.1. Comparativa entre modelo web clásico y modelo AJAX . . . . .	15
3.1. Casos de uso de la aplicación . . . . .	18
4.1. Estructura global . . . . .	21
4.2. HTMLView . . . . .	21
4.3. Actions . . . . .	23
4.4. Front Controller . . . . .	30
4.5. Paquete model . . . . .	32
4.6. Transfer Objects . . . . .	34
4.7. Entities . . . . .	35
4.8. User Facade . . . . .	39
4.9. User Facade (2) . . . . .	40
4.10. Search Facade . . . . .	42
4.11. Search Facade (2) . . . . .	43
4.12. Search Facade (3) . . . . .	44

5.1. Patrón Business Delegate . . . . .	48
5.2. Patrón Front Controller . . . . .	51
6.1. Pantalla principal usuario no autenticado . . . . .	54
6.2. Pantalla de autenticación . . . . .	55
6.3. Pantalla de registro . . . . .	57
6.4. Pantalla principal usuario autenticado . . . . .	58
6.5. Pantalla de actualización de perfil . . . . .	59
6.6. Pantalla de cambio de contraseña . . . . .	60
6.7. Pantalla principal (opciones WMSs) . . . . .	62
6.8. Búsqueda de WMSs . . . . .	63
6.9. Búsqueda de WMSs por nombre . . . . .	64
6.10. Pantalla de “Mis WMSs” . . . . .	65
6.11. Pantalla de actualización de WMS . . . . .	67
6.12. Pantalla de detalles de WMS . . . . .	68
6.13. Pantalla de añadir WMS . . . . .	69
6.14. Pantalla de añadir mapa . . . . .	70
6.15. Pantalla selector zona geográfica . . . . .	72
6.16. Pantalla de selección de WMS . . . . .	73
6.17. Pantalla de selección de capas . . . . .	74
6.18. Búsqueda de mapas . . . . .	76
6.19. Búsqueda de mapas por nombre . . . . .	77
6.20. Búsqueda de mapas por etiquetas (tags) . . . . .	78
6.21. Búsqueda de mapas por zona geográfica . . . . .	79
6.22. Pantalla de “Mis mapas” . . . . .	80

6.23. Pantalla de detalles de mapa . . . . .	82
6.24. Pantalla de actualización de mapa . . . . .	83
6.25. Pantalla de visualización de mapa . . . . .	84
6.26. Pantalla de añadir POI (punto de interés) . . . . .	86
6.27. Pantalla principal (opciones POIs) . . . . .	87
6.28. Búsqueda de POIs (puntos de interés) . . . . .	89
6.29. Búsqueda de POIs (puntos de interés) por nombre . . . . .	90
6.30. Búsqueda de POIs (puntos de interés) por etiquetas (tags) . .	91
6.31. Búsqueda de POIs (puntos de interés) por zona geográfica . .	92
6.32. Pantalla de “Mis POIs” . . . . .	93
6.33. Pantalla de actualización de POI (punto de interés) . . . . .	94
6.34. Pantalla de detalles de POI (punto de interés) . . . . .	96
6.35. Pantalla principal (opciones usuarios) . . . . .	97
6.36. Búsqueda de usuarios . . . . .	98
6.37. Búsqueda de usuarios por login (identificador de usuario) . . .	99



## 1.1. Motivación

Se trata de crear una aplicación web participativa donde los usuarios que estén registrados puedan crear sus mapas personalizados a partir de la información obtenida de servidores WMS ajenos a dicha aplicación. Además debe permitirles añadir a dichos mapas puntos de información que puedan ser visualizados por los demás usuarios que lo deseen.

La aplicación también debe permitir que un usuario que no esté registrado pueda buscar y visualizar los objetos creados por los usuarios registrados, aunque no pueda luego almacenarlos ni crear los suyos propios a no ser que se registre.

Hemos mencionado que la información para crear los mapas es obtenida a partir de servidores WMS, pero ¿qué es un servidor WMS? Es un servicio web que genera cartografía a partir de información geográfica. A través de una petición HTTP se puede indicar las capas a visualizar, los estilos a utilizar, el formato y la resolución de la imagen. Esta definición se ampliará posteriormente en el apartado de *Estándares y tecnologías utilizadas*.

## 1.2. Objetivos del proyecto

- **Modelado según el paradigma de la orientación a objetos:** Aunque los diagramas producidos durante la fase de diseño no formen parte del software en el sentido más estricto de la palabra, no dejan de ser una parte importante del proyecto. El presente documento contiene, por tanto, los diagramas de aquellas partes de la aplicación de vital relevancia o especial dificultad. Los diagramas utilizan el lenguaje de modelado unificado UML.
- **Utilización de patrones de diseño:** El diseño de la aplicación ha de ser flexible y adoptar soluciones que se han utilizado con éxito en el pasado ante los problemas más comunes. La mayor parte de los problemas deben solucionarse empleando patrones de diseño desde la arquitectura global de la aplicación hasta los problemas más puntuales de creación de objetos y de comportamiento de los mismos.
- **Persistencia:** Las acciones que lleven a cabo los usuarios y aquellas que el sistema realice por cuenta propia han de quedar guardadas de forma que pueda recuperarse el sistema en un estado actual y consistente ante cualquier problema.
- **Capacidad de funcionamiento distribuido:** La aplicación debe poder distribuir sus responsabilidades en diferentes máquinas. Por ejemplo, se puede instalar el servidor de bases de datos y la parte del modelo de la aplicación en sendas máquinas a las que la aplicación web accederá de forma remota.

- **Posibilidad de accesos concurrentes y escalabilidad:** No sólo se desea que varios usuarios puedan interactuar a la vez con el sistema, sino que la aplicación ha de funcionar de forma eficaz y eficiente aunque el número de usuarios crezca aumentando el número de accesos concurrentes al mismo.

### 1.3. Estructura de la memoria

En el siguiente apartado de la memoria se explican en detalle los estándares y tecnologías utilizadas en la realización del proyecto.

A continuación se detallan los procesos de Análisis, Diseño e Implementación, explicando las decisiones tomadas para la elaboración del proyecto, al mismo tiempo que ofrece una visión detallada de las mismas.

Posteriormente se hace una breve descripción de los patrones utilizados en el proyecto, para continuar luego con un manual de usuario que detalla cómo realizar las distintas acciones que puede realizar un usuario (ya esté registrado en la aplicación o no).

Por último se incluye un apartado donde se muestran las líneas de mejora que se pueden seguir en un futuro para mejorar la aplicación.

La memoria también consta de una serie de apéndices que explican como instalar la aplicación (Manual de compilación e instalación), explican el contenido del CD adjunto (Contenido del CD adjunto), hacen mención al código licenciado usado para la realización del proyecto (Uso de código licenciado) y por último muestran un glosario con algunos acrónimos y términos utilizados en esta memoria.



## Estándares y tecnologías utilizadas

En este capítulo se realiza un breve resumen de los principales estándares y tecnologías que se han utilizado en el desarrollo de la aplicación.

### 2.1. Web Map Service

El Web Map Service (WMS) define un servicio web que genera cartografía a partir de información geográfica. A través de una petición HTTP se puede indicar las capas a visualizar, los estilos a utilizar, el formato y la resolución de la imagen.

Para llevar a cabo esta funcionalidad, cualquier WMS debe soportar las siguientes operaciones:

- **GetCapabilities:** devuelve información del servicio ofrecido.
- **GetMap:** encargado de construir y devolver un mapa como una imagen.
- **GetFeatureInfo:** esta operación devuelve información acerca de los objetos representados en un píxel de la imagen.

De acuerdo con estas operaciones básicas se definen dos tipos de servicios:

- **WMS Básico:** los estilos están predefinidos y no se pueden cambiar, lo que implica que de los estilos sólo se conozca el nombre pero no su definición y los clientes del servicio no puedan definir sus propios estilos para la información geográfica.
- **WMS con SLD:** soluciona los problemas del tipo anterior gracias a la definición del lenguaje SLD lo que le permite:
  - Utilizar este lenguaje como una biblioteca.
  - Utilizar el lenguaje para definir nuevos estilos.
  - Implementar un servicio WMS en cascada.
  - Ampliar las funcionalidades de las operaciones básicas:
    - *GetCapabilities:* proporciona información adicional referente a la funcionalidad SLD soportada.
    - *GetMap:* posibilita indicar los estilos que un usuario define.
  - Permite definir operaciones nuevas:
    - *DescribeLayer:* devuelve información acerca de las capas definidas en el WMS.
    - *GetLegendGraphic:* devuelve el ícono que representa el estilo.
    - *GetStyles:* para consultar los estilos.
    - *UpdateStyles:* permite modificar los estilos definidos.

## 2.2. J2EE

El estándar J2EE proporciona un conjunto de especificaciones de APIs Java para la construcción de aplicaciones empresariales. Por tanto, y dado

que sólo se trata de un conjunto de especificaciones, una aplicación construida en J2EE no depende de una implementación particular, por lo que se favorece la posibilidad de utilizar tecnologías de distintos fabricantes para la ejecución de la misma, lo que proporciona una, más que deseable, independencia del fabricante de la tecnología de soporte.

Debido a la utilización de la plataforma J2EE, para la codificación de la aplicación se ha utilizado el lenguaje de programación Java. El uso de Java y J2EE simplifica el software a través del uso de componentes modulares y estándar, sólidamente probados, proveyendo un conjunto de servicios a los mismos.

El uso de estos componentes permite crear aplicaciones escalables, confiables e integradas con los sistemas existentes en tiempos cada vez más cortos.

### 2.3. EJB3

EJB, Enterprise JavaBean, es una tecnología J2EE para la implementación de la capa modelo de una aplicación proporcionando:

- soporte para persistencia
- soporte para implementación de fachadas (ocultación de APIs de transacciones y seguridad)

Las versiones anteriores (EJB 1.x y 2.x) fueron criticadas debido a que eran difíciles de usar y que a pesar de que el soporte para persistencia permitía programar de manera independiente del tipo de BD (relacional, objetual...) era poco potente debido a que no se permitía realizar muchos tipos de con-

sultas usuales (join, group by...), borrados o actualizaciones en masa, etc. A consecuencia de estas complejidades y deficiencias surgieron diversos proyectos encaminados a resolverlos: Hibernate, TopLink, JDO o Spring son ejemplos de ello, con mayor o menor grado de éxito y aceptación.

Precisamente inspirado en alguno de estos proyectos anteriores (concretamente en Hibernate, ToLink y JDO) surge EJB 3.0, que puede ser visto como un mapeador objeto/relacional (en consecuencia no oculta que el tipo de base de datos es relacional), incluye el nuevo API de persistencia de JAVA, proporciona un modelo más sencillo para la implementación de fachadas y puede usarse en entornos J2SE (fuera de un contenedor J2EE) y sin el resto de componentes de EJB, lo que permite que sea válido para aplicaciones *standalone*.

Consta de:

- **Anotaciones:** el desarrollador puede anotar las clases persistentes (entidades) para que la implementación del API de persistencia sepa cómo mapear las instancias a la base de datos (por ejemplo, indicando el nombre de la tabla, los nombres de las columnas a las que mapear los atributos, etc.).
- El API propiamente dicho: correspondiente a *javax.persistence* cuyos objetos principales son:
  - *EntityManager*: que permite crear, encontrar por clave primaria y eliminar objetos persistentes, y crear objetos *Query*.
  - *Query*: permite lanzar las consultas en EJB-QL.

Internamente las implementaciones usan el API de JDBC (transparente al programador).

- **EJB-QL:** acrónimo de EJB Query Language. Es un lenguaje de consultas de búsqueda y borrados/actualizaciones en masa, siendo la implementación la que traduce al SQL de la base de datos que se esté usando.

Finalmente comentar que las fachadas podrán ser locales o remotas, ocultando los APIs de transacciones y seguridad al programador, pudiéndose distinguir dos tipos de fachadas:

- **SessionBeans:** que constituyen las fachadas “normales”(operaciones síncronas), teniendo que definir una interfaz y una implementación. Dentro de estas fachadas tendremos:
  - *Stateless Session Beans (SLSB)*: fachadas sin estado.
  - *Stateful Session Beans (SFSB)*: fachadas con estado.
- **Message-Driven Beans (MDB)**: son fachadas del modelo que implementan casos de uso por los que el cliente no puede esperar a que terminen, funcionando bajo el paradigma del envío de mensajes (operaciones asíncronas).

## 2.4. Struts

Struts es un “framework open source”, facilitado por Apache, para construir aplicaciones web en el mundo Java. Se basa en tecnologías estándar Java y su arquitectura se ajusta al patrón arquitectónico “Model-View-Controller”.

La separación estricta de las capas del patrón permite la integración de diferentes tipos de tecnologías en cada una de ellas.

Este “framework” ofrece cierta funcionalidad para implementar aplicaciones multilenguaje, permitiendo definir las páginas de una aplicación, además de sus correspondientes acciones asociadas, de una manera abstracta y externalizada, lo que permite separar la lógica de control de la lógica de la vista.

Toda esta información se mantiene a través de XML, en un fichero de configuración, lo que permite un eficiente y fácil mantenimiento del nivel de presentación de las aplicaciones. Además, posibilita el ser complementado por medio de “plug-ins” para poder incrementar su funcionalidad, como por ejemplo, en el caso del manejo de peticiones bajo el protocolo HTTPS, para permitir el envío seguro de información entre el cliente y el servidor.

A nivel de la vista de la aplicación, proporciona un conjunto muy completo de librerías de etiquetas, con lo que las JSPs quedan bastante limpias de código Java.

Por otra parte, el controlador se puede implementar siguiendo un mapeo abstracto de los elementos de presentación de la aplicación (formularios de datos, acciones... ) según el patrón Front Controller. De esta forma el controlador estará totalmente desacoplado de la lógica de negocio.

En lo referente al modelo, Struts no ofrece ninguna funcionalidad en la capa de acceso a datos, por lo que es posible el empleo de cualquier tecnología, haciéndose uso en este caso de EJB3, mucho más sencillo que JDBC.

## 2.5. XML

Es una tecnología estandarizada por el “World Wide Web Consortium” (W3C), que se basa en documentos de texto plano en los que se utilizan etiquetas para delimitar los elementos de un documento. Estas etiquetas son definidas en función del tipo de datos que está describiendo y no de la apariencia final que tendrán en pantalla o en la copia impresa, como ocurría en HTML. XML permite, además, definir nuevas etiquetas y ampliar las existentes.

## 2.6. CSS (Cascade Style Sheet)

Es una especificación, estandarizada por el W3C, sobre los estilos físicos aplicables a un documento HTML, que fue concebida tratando de proporcionar la separación definitiva entre la estructura y la presentación del documento. Entre las muchas ventajas del uso de CSS, conviene destacar las siguientes:

- **Flexibilidad:** Si se hace uso de hojas de estilo externas, todo el diseño gráfico de la web se encuentra centralizado en un único punto, por lo que realizar cambios en la vista de la aplicación es relativamente sencillo.
- **Rapidez:** Al existir menos código en las páginas, éstas son descargadas más rápidamente por los usuarios, con lo que se gana eficiencia.
- **Dinamismo:** Aunque CSS presenta algunas características que dotan de cierto dinamismo a algunos objetos, es fácilmente combinable con

JavaScript, lo que permite realizar un diseño con cualquier grado de dinamismo deseado.

## **2.7. JSP (Java Server Pages)**

Una página JSP es un tipo especial de “servlet” orientado a generar el texto de la interfaz gráfica. Este tipo de páginas necesitan de un programa que las contenga, envíe efectivamente páginas web al servidor, y reciba las peticiones, las distribuya entre los “servlets” y lleve a cabo todas las tareas de gestión propias de un servidor web. Este programa es el denominado servidor de aplicaciones o contenedor web.

La primera vez que se accede a una página JSP el servidor de aplicaciones genera un servlet a partir de la página JSP, lo compila y lo carga en memoria. Si no es la primera vez, le pasa la petición al “servlet” anteriormente generado. Una página JSP es también una página web por lo que permite la inclusión de código HTML, pero además también permite añadir código Java, aunque no es una buena práctica porque no permite una separación clara de roles, es decir, el diseñador gráfico debería conocer entonces Java y saber programar para diseñar la página, lo cual no es una buena opción. Para solucionarlo se hace uso de librerías de tags JSP que permiten realizar diversas acciones sobre los elementos de una página web, tales como iteración sobre colecciones, internacionalización de mensajes, acceso a documentos XML....

Además de la librería de tags anteriormente mencionada proporcionada por Struts, se hace uso en esta aplicación de JSTL (JSP Standard Tag Library).

## 2.8. AJAX

Acrónimo de Asynchronous JavaScript And XML (JavaScript y XML asíncronos, donde XML es un acrónimo de eXtensible Markup Language).

Es una técnica de desarrollo web para crear aplicaciones interactivas. Éstas se ejecutan en el cliente, es decir, en el navegador del usuario, y mantiene comunicación asíncrona con el servidor en segundo plano. De esta forma es posible realizar cambios sobre la misma página sin necesidad de recargarla. Esto significa aumentar la interactividad, velocidad y usabilidad en la misma. AJAX es una combinación de tecnologías ya existentes:

- **XHTML (o HTML)** y **hojas de estilos en cascada**(CSS) para el diseño que acompaña a la información.
- **Document Object Model** (DOM) accedido con un lenguaje de scripting por parte del usuario, especialmente implementaciones ECMAScript como JavaScript y JScript, para mostrar e interactuar dinámicamente con la información presentada.
- El objeto **XMLHttpRequest** para intercambiar datos asíncronamente con el servidor web. En algunos frameworks y en algunas situaciones concretas, se usa un objeto iframe en lugar del XMLHttpRequest para realizar dichos intercambios.
- **XML** es el formato usado comúnmente para la transferencia de vuelta al servidor, aunque cualquier formato puede funcionar, incluyendo HTML preformatteado, texto plano, JSON y hasta EBML.
- JavaScript para unir todo lo anterior.

AJAX no constituye una tecnología en sí, sino que es un término que engloba a un grupo de éstas que trabajan conjuntamente.

El modelo clásico de aplicaciones web funciona de esta forma: la mayoría de las acciones del usuario en la interfaz disparan un requerimiento HTTP al servidor web. El servidor efectúa un proceso (recopila información, procesa números, hablando con varios sistemas propietarios), y le devuelve una página HTML al cliente. Este es un modelo adaptado del uso original de la Web como un medio hipertextual.

Una aplicación AJAX elimina la naturaleza de “arrancar-frenar-arrancar-frenar” de la interacción en la Web introduciendo un intermediario -un motor AJAX- entre el usuario y el servidor. Este motor AJAX permite que la interacción del usuario con la aplicación suceda asíncronamente (independientemente de la comunicación con el servidor). Así el usuario nunca estará mirando una ventana en blanco del navegador y un icono de reloj de arena esperando a que el servidor haga algo.

Se observa como el concepto de AJAX es el de cargar y renderizar una página, luego mantenerse en esa página mientras scripts y rutinas van al servidor buscando, en background, los datos que son usados para actualizar la página, lo que permite mostrar u ocultar porciones de la misma.

Ejemplos de compañías que están usando AJAX son el motor de búsqueda de Amazon, que aplica tecnologías similares, o Google, que está haciendo una significativa inversión en su acercamiento a AJAX: todos los grandes productos que Google ha introducido últimamente (Orkut, Gmail, Google Groups, Google Suggest o Google Maps) son aplicaciones AJAX.

En la figura 2.1 se observa una comparativa entre ambos modelos.

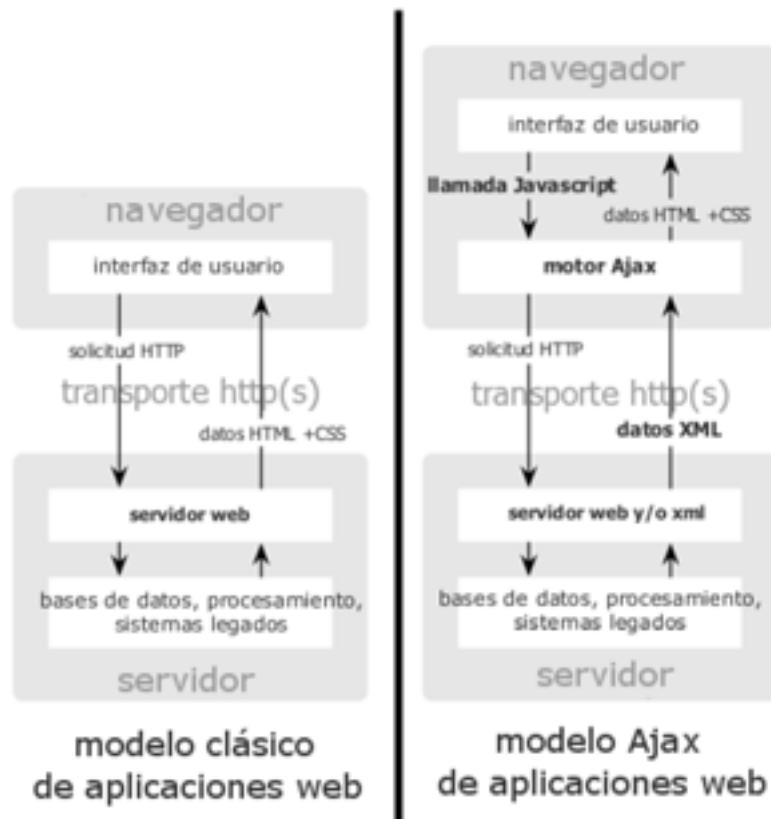


Figura 2.1: Comparativa entre modelo web clásico y modelo AJAX



# 3

## Análisis

En este capítulo se procederá a la descomposición del dominio en el que trabaja la aplicación. De esta manera, se podrán identificar los distintos casos de uso que debe abordar la misma para cubrir las distintas necesidades de los usuarios.

Es necesario distinguir previamente dos tipos de perfil de usuario que tendrán a su disposición distintas funcionalidades del sistema. Estos dos tipos son el usuario no autenticado y el usuario autenticado, los cuales difieren únicamente en que el segundo está vinculado a una cuenta de usuario y dispone, por tanto, de todas las operaciones relativas a la cuenta.

En la figura 3.1 se muestran los distintos casos de uso que se identificaron en la aplicación.

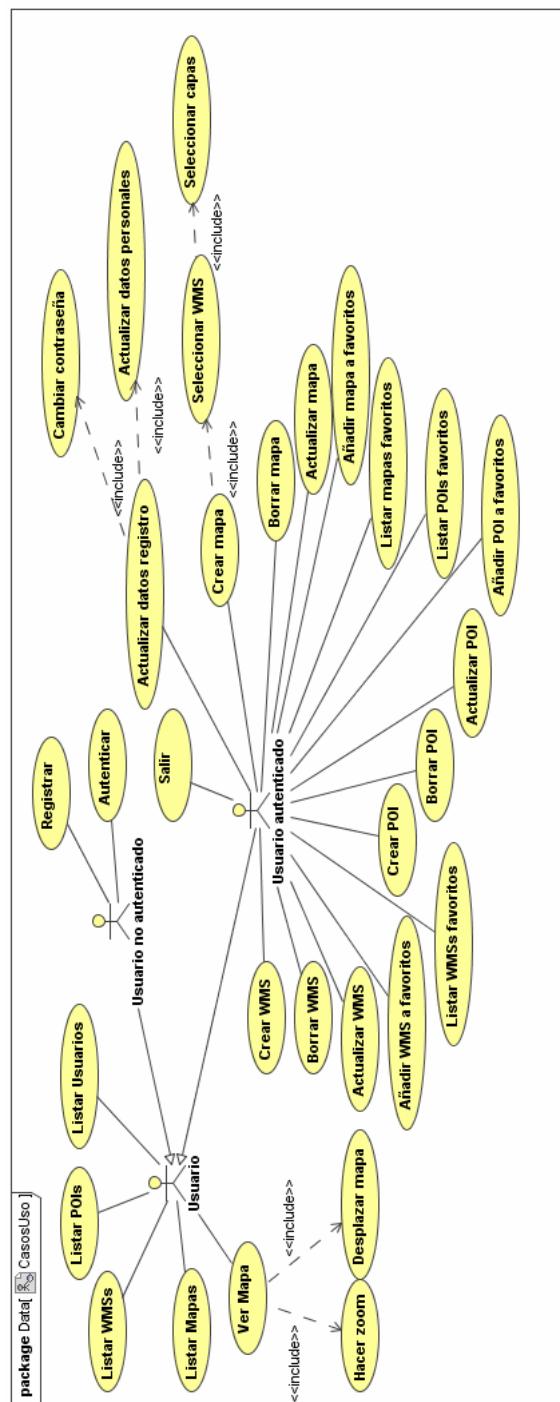


Figura 3.1: Casos de uso de la aplicación

# 4

## Diseño e implementación

Una vez descompuesto el dominio y analizadas las necesidades del usuario, el siguiente paso es modelar el sistema. Durante la fase de diseño se procede al modelado de la forma del sistema, de manera que proporcione soporte a los requisitos encontrados en la fase anterior. En concreto, se busca crear una entrada apropiada para actividades de implementación subsiguientes, descomponiendo los trabajos de la misma en partes más manejables.

Por su parte, durante la fase de implementación se codifica el sistema en términos de componentes, es decir, ficheros de código, scripts, ejecutables, etc. Además, también se realiza una distribución del sistema asignando componentes ejecutables a nodos en un diagrama de despliegue.

Para llevar a cabo este trabajo se hace uso de una serie de artefactos, denominados patrones. Estos artefactos proporcionan una solución a un problema de diseño no trivial que es efectiva y reusable. Además, para la representación del modelo de diseño se hace uso de los diagramas UML: vista estática de clases y diagramas de secuencia, sobre todo.

La aplicación se ha diseñado en una estructura de subsistemas. Actualmente existe un único subsistema principal llamado MapViewer, pero se ha elegido

la mencionada estructura ya que en el futuro el sistema puede ser modificado y ampliado por diversos desarrolladores. Está previsto que puedan aparecer subsistemas que complementen al principal, sean o no aplicaciones web.

Como se podrá apreciar en breve, el subsistema MapViewer está distribuido en capas según los patrones arquitectónicos Layers (Capas) y MVC o Model-View-Controller (Modelo-Vista-Controlador), que se explican en detalle en el apartado de *Patrones utilizados*. De esta manera se logra una independencia entre las mismas, de forma que cualquier cambio en una de ellas no debería afectar a las otras, a no ser que la interfaz que separa a ambas cambie. Ésta es la base para un diseño flexible y desacoplado.

## **4.1. Arquitectura global**

En la figura 4.1 se puede ver la estructura global de la aplicación. A continuación se comentarán en detalle cada una de las partes. También se incluyen varios diagramas de clases para acompañar la aplicación. Dado que algunos son demasiados grande y tuvieron que ser reducidos, se pueden ver a tamaño real o bien en formato PDF (aumentando el zoom), en formato MagicDraw (en la carpeta *Diagramas* dentro del código fuente de la aplicación) o bien en formato de imagen PNG (en la carpeta *Imágenes* dentro de la carpeta de la memoria).

## **4.2. Paquete EJBConfiguration**

Aquí se encuentran los ficheros de configuración de EJB3 para su correcto funcionamiento bajo JBoss.

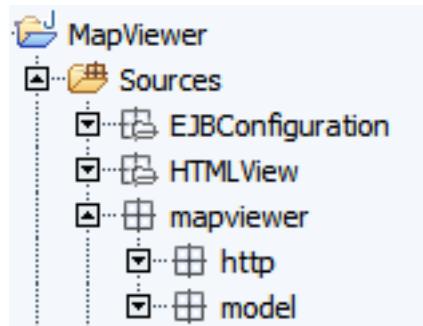


Figura 4.1: Estructura global

### 4.3. Paquete HTMLView

Contiene los ficheros necesarios para la interfaz gráfica de la aplicación web. Su estructura se puede observar en la figura 4.2.

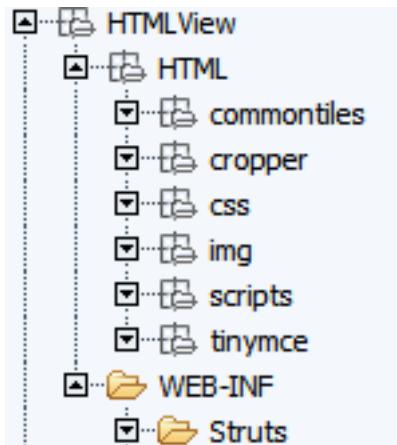


Figura 4.2: HTMLView

#### 4.3.1. HTML

En este paquete se sitúan todas las páginas jspx usadas para confeccionar la aplicación; los ficheros necesarios para el uso del patrón Tiles, que nos permite repetir una misma estructura en las páginas que se muestran (encabezado, pie de página y barra lateral) de forma que en cada página sólo

tengamos que incluir en la parte de contenido la página jspx que queramos mostrar (la carpeta *common tiles* y la subcarpeta *layouts*); una hoja de estilos CSS que nos ayuda a dar el aspecto visual que queramos a la aplicación (bajo la carpeta *css*); las imágenes que usa la aplicación (en la carpeta *img*); y varios ficheros JavaScript con las funcionalidades necesarias para la correcta interacción del usuario con la aplicación web (incluidos los scripts realizados por personas ajenas y que se describen en el apartado *Uso de código licenciado*).

#### 4.3.2. WEB-INF

Incluye los ficheros de configuración de Struts, Tiles y Validator y el descriptor de la aplicación *web.xml*.

### 4.4. Paquete http.controller

Se encarga de modelar la capa controlador de la aplicación web y contiene un conjunto de clases *Action*, que interactúan con el modelo y seleccionan la siguiente vista.

#### 4.4.1. Paquete actions

Contiene las anteriormente citadas acciones que extienden a la clase *DefaultAction* definiendo para ello el método *doExecute()*. Se implementan las siguientes acciones que se pueden ver en la figura 4.3:

- **AddInterestingMapAction:** añade un mapa a la lista de mapas que le interesan al usuario autenticado.

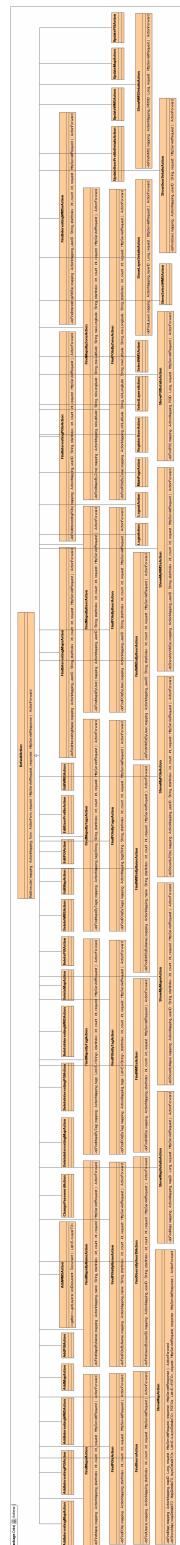


Figura 4.3: Actions

- **AddInterestingPOIACTION:** añade un POI (punto de interés) a la lista de POIs que le interesan al usuario autenticado.
- **AddInterestingWMSACTION:** añade un servidor WMS a la lista de WMSs que le interesan al usuario autenticado.
- **AddMapACTION:** crea un mapa y lo añade a la base de datos.
- **AddPOIACTION:** crea un POI (punto de interés) y lo añade a la base de datos.
- **AddWMSACTION:** crea un servidor WMS (con todas sus capas y estilos de capas) y lo añade todo a la base de datos.
- **ChangePasswordACTION:** cambia la contraseña del usuario.
- **DeleteInterestingMapACTION:** elimina un mapa de la lista de mapas que le interesan al usuario autenticado.
- **DeleteInterestingPOIACTION:** elimina un POI (punto de interés) de la lista de POIs que le interesan al usuario autenticado.
- **DeleteInterestingWMSACTION:** elimina un servidor WMS de la lista de WMSs que le interesan al usuario autenticado.
- **DeleteMapACTION:** elimina un mapa de la base de datos.
- **DeletePOIACTION:** elimina un POI (punto de interés) de la base de datos.
- **DeleteWMSACTION:** elimina un servidor WMS de la base de datos.

- **EditMapAction:** prepara los datos para crear o actualizar un mapa en la base de datos según los parámetros que reciba.
- **EditPOIACTION:** prepara los datos para crear o actualizar un POI (punto de interés) en la base de datos según los parámetros que reciba.
- **EditUserProfileAction:** prepara los datos para crear o actualizar un usuario en la base de datos según los parámetros que reciba.
- **EditWMSAction:** prepara los datos para crear o actualizar un servidor WMS en la base de datos según los parámetros que reciba.
- **FindInterestingMapsAction:** muestra una lista con los mapas que le interesan al usuario autenticado.
- **FindInterestingPOIsAction:** muestra una lista con los POIs (puntos de interés) que le interesan al usuario autenticado.
- **FindInterestingWMSSsAction:** muestra una lista con los servidores WMSS que le interesan al usuario autenticado.
- **FindMapsAction:** muestra una lista con todos los mapas.
- **FindMapsByNameAction:** muestra una lista con todos los mapas que tienen un nombre igual a uno dado.
- **FindMapsByTagAction:** muestra una lista con todos los mapas que tienen, al menos, una etiqueta (tag) dada.
- **FindMapsByTagsAction:** muestra una lista con todos los mapas que tienen, al menos, las etiquetas (tags) dadas.

- **FindMapsByUserAction:** muestra una lista con todos los mapas de un usuario dado.
- **FindMapsByZoneAction:** muestra una lista con todos los mapas de una zona dada.
- **FindPOIsAction:** muestra una lista con todos los POIs (puntos de interés).
- **FindPOIsByNameAction:** muestra una lista con todos los POIs (puntos de interés) que tienen un nombre igual a uno dado.
- **FindPOIsByTagAction:** muestra una lista con todos los POIs (puntos de interés) que tienen, al menos, una etiqueta (tag) dada.
- **FindPOIsByTagsAction:** muestra una lista con todos los POIs (puntos de interés) que tienen, al menos, las etiquetas (tags) dadas.
- **FindPOIsByUserAction:** muestra una lista con todos los POIs (puntos de interés) de un usuario dado.
- **FindPOIsByZoneAction:** muestra una lista con todos los POIs (puntos de interés) de una zona dada.
- **FindUsersAction:** muestra una lista con todos los usuarios.
- **FindUsersByUserIDAction:** muestra una lista con todos los usuarios con un identificador dado. En principio, debería devolver uno o ninguno, nunca más de uno.
- **FindWMSSsAction:** muestra una lista con todos los servidores WMS.

- **FindWMSsByNameAction:** muestra una lista con todos los servidores WMS que tienen un nombre igual a uno dado.
- **FindWMSsByUserAction:** muestra una lista con todos los servidores WMS de un usuario dado.
- **LoginAction:** permite a un usuario registrado entrar en la aplicación.
- **LogoutAction:** permite a un usuario autenticado salir de la aplicación.
- **MainPageAction:** se usa para establecer las opciones que aparecen en la barra lateral.
- **RegisterUserAction:** permite a un usuario registrarse en la aplicación.
- **SelectLayersAction:** permite elegir las capas que se desean incluir en un mapa.
- **SelectWMSAction:** permite seleccionar un servidor WMS para añadir alguna de sus capas en un mapa.
- **ShowLayerDetailsAction:** muestra los detalles de una capa dada.
- **ShowMapAction:** muestra un mapa dado.
- **ShowMapDetailsAction:** muestra los detalles de un mapa dado.
- **ShowMyMapsAction:** muestra una lista con todos los mapas del usuario autenticado.
- **ShowMyPOIsAction:** muestra una lista con todos los POIs (puntos de interés) del usuario autenticado.

- **ShowMyWMSSAction:** muestra una lista con todos los servidores WMS del usuario autenticado.
- **ShowPOIDetailsAction:** muestra los detalles de un POI (punto de interés) dado.
- **ShowSelectWMSAction:** prepara los datos para elegir un servidor WMS para añadir alguna de sus capas en un mapa.
- **ShowUserDetailsAction:** muestra los detalles de un usuario dado.
- **ShowWMSDetailsAction:** muestra los detalles de un servidor WMS dado.
- **UpdateMapAction:** actualiza un mapa en la base de datos.
- **UpdatePOIAction:** actualiza un POI (punto de interés) en la base de datos.
- **UpdateUserProfileDetailsAction:** actualiza los detalles de un usuario en la base de datos.
- **UpdateWMSAction:** actualiza un servidor WMS en la base de datos.

#### 4.4.2. Paquete frontcontroller

Permite la delegación del procesamiento de peticiones por parte del *FrontController* de *Struts*(**ActionServlet**) en la clase RequestProcessor, de manera que su método principal(**process()**) actúe como método plantilla que se implementa en términos de otros.

Su implementación siguiendo el patrón *Cadena de Responsabilidad* nos permite asegurar que un usuario tenga en su sesión todo lo que necesita, controlar que tengamos cargado todo el entorno necesario para la aplicación y filtrar aquellas llamadas a actions que precisan que el usuario esté autenticado. Su estructura puede verse en la figura 4.4.

#### 4.4.3. Paquete session

Fachada que facilita la implementación de las acciones en el controlador, dotándole de operaciones por cada operación que necesite actualizar la sesión y/o las cookies.

#### 4.4.4. Paquete util

Contiene una clase utilidad para comparar objetos del tipo (*etiqueta, valor*).

### 4.5. Paquete http.util

Contiene una clase utilidad para instanciar objetos del tipo (*etiqueta, valor*).

### 4.6. Paquete http.view.messages

Se incluyen los ficheros de mensajes de los idiomas soportados por la aplicación (por el momento sólo el español). Es importante hacer notar que lo que se internacionaliza en estos ficheros es toda aquella información que no se trae directamente de la base de datos. El fichero de mensajes a utilizar será

## 4.6. Paquete http.view.messages

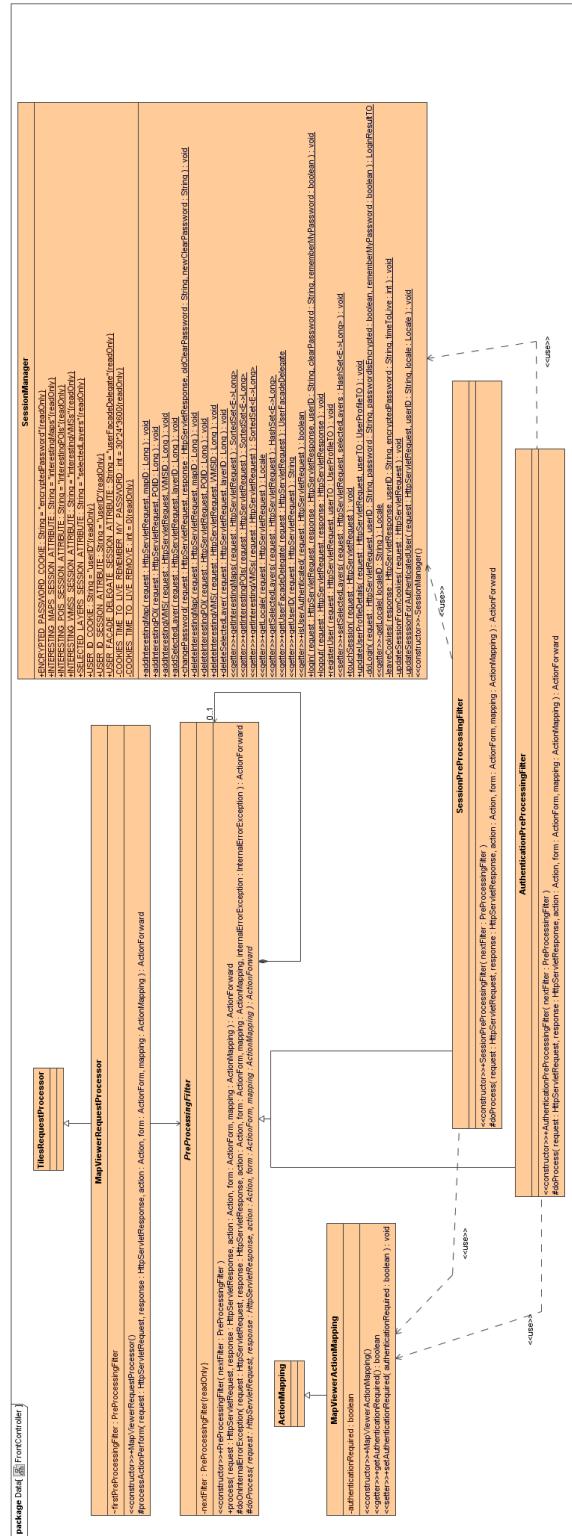


Figura 4.4: Front Controller

el del idioma que tenga configurado el navegador que esté usando el propio usuario; en caso de no estar soportado por la aplicación web se utilizará el idioma por defecto (el español). Los mensajes en los ficheros están ordenados por la clave alfabéticamente para facilitar su manejo.

#### 4.7. Paquete model

En este paquete se encuentra el modelo de datos de la aplicación y todas las funcionalidades del sistema que tengan que relacionarse con los datos que se estén utilizando tendrán que utilizar las operaciones proporcionadas por las clases de este paquete. La estructura del paquete se puede ver en la figura 4.5.

La aplicación web consta de dos fachadas:

- **searchFacade**: modela las operaciones relativas a la búsqueda de objetos. Se trata de una fachada sin estado (*Stateless*).
- **userFacade**: modela el resto de operaciones que puede realizar un usuario. Se trata de una fachada con estado (*Stateful*).

Cada una de estas fachadas tendrá asociados unos *Transfer Object* para el manejo de la información obtenida con sus respectivos *Chunk* en el supuesto de que el caso de uso dé soporte al patrón *Page-by-page*(todos ellos bajo el subpaquete TO de cada fachada).

Por otra parte tendremos las clases persistentes (*entity*) que dan soporte a la aplicación con sus correspondientes Transfer Object.

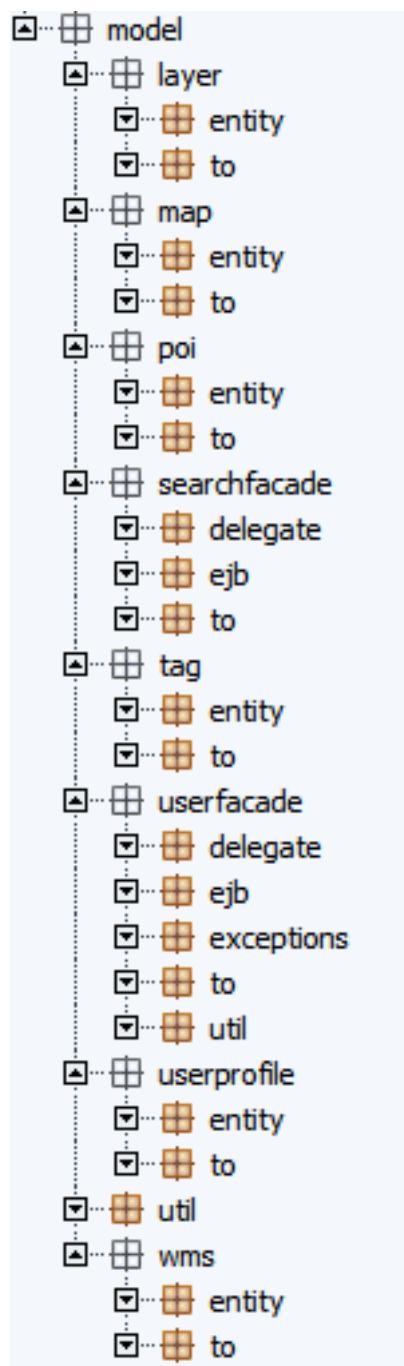


Figura 4.5: Paquete model

#### 4.7.1. Transfer Object

Su finalidad es agrupar un conjunto de atributos que se corresponden con los de un objeto del dominio. Se han definido los siguientes:

- **LayerTO:** modela una capa de un servidor WMS.
- **MapTO:** modela un mapa.
- **POITO:** modela un POI (punto de interés).
- **TagTO:** modela una etiqueta (tag) de un mapa o de un POI.
- **UserProfileTO:** modela los datos no personales de un usuario tales como el login y la contraseña.
- **UserProfileDetailsTO:** modela los datos personales de un usuario.
- **WMSTO:** modela un servidor de WMS.

Todos ellos constan de los correspondientes métodos *getXXX()* / *setXXX()* para acceder a los atributos o para modificarlos. También redefinen los métodos *toString()* para facilitar la depuración de la aplicación y *equals()* para realizar correctamente comparaciones entre dos objetos TO.

#### 4.7.2. Entities

Son las clases persistentes definidas con la anotación *@Entity*. Se ha definido un entity para cada uno de los *Transfer Object* comentados con anterioridad.

Como clase persistente, ha de cumplir una serie de requisitos:

- Puede extender a otra.

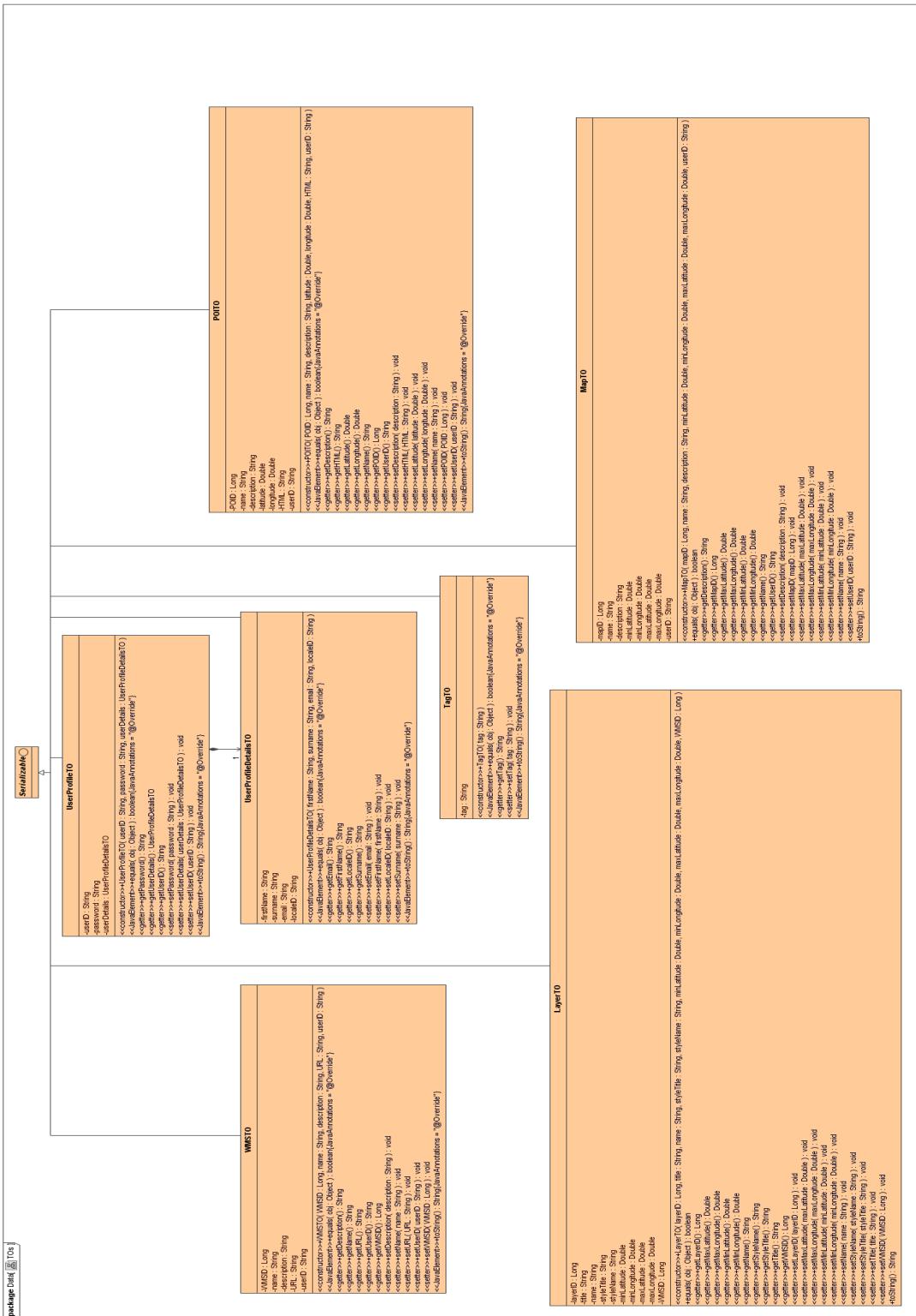


Figura 4.6: Transfer Objects

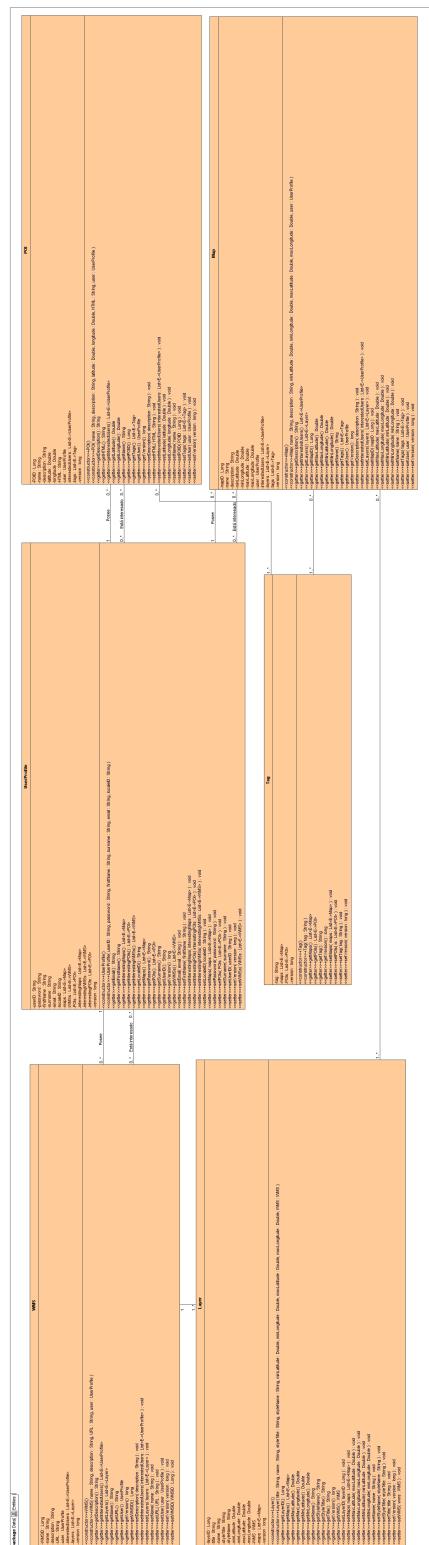


Figura 4.7: Entities

- Puede ser abstracta.
- Puede tener una relación de asociación con otra.
- Estado persistente: definido con atributos de visibilidad no pública (privada, protegida o de paquete) y con métodos públicos get/set para acceder y modificar los valores. La implementación de EJB accede al estado directamente a través de sus atributos (tipo de acceso: campo) o mediante método get/set (tipo de acceso: propiedad).
- Los atributos o propiedades pueden ser principalmente: tipos primitivos y sus contrapartidas objetuales, enumerables (se mapean a una columna) y colecciones de entidades (relaciones) que se mapean a columnas y/o tablas dependiendo de la cardinalidad y direccionalidad de la relación.

Internamente cada *entity* tendrá un constructor público sin argumentos no declarado como *final*, las notaciones *@Table* y *@Column* en caso de que el nombre de la tabla y/o columna en base de datos no se corresponda con el nombre del *entity* que lo referencia y un atributo anotado con la notación *@Id* que lo identifica como clave primaria en la base de datos (y la secuencia utilizada para generar los identificadores, en caso de que sea necesario realizar inserciones en la tabla). Finalmente cada *entity* tendrá definidas una serie de relaciones y direccionalidades con otros entities, que pueden ser de los siguientes tipos:

- OneToOne
- OneToMany

- ManyToOne
- ManyToMany

En la figura que muestra los entities también se pueden observar las distintas relaciones que se definieron entre ellos junto con su cardinalidad. Todas las relaciones se definieron bidireccionales pensando en su uso en futuras mejoras de la aplicación, ya que, en la actualidad, algunas de ellas sólo se navegan en una de las direcciones.

#### 4.7.3. UserFacade

Es una fachada con estado creada para implementar los casos de uso relativos a las operaciones que puede realizar un usuario autenticado en la aplicación. El esquema seguido para la implementación de las fachadas de la aplicación es el siguiente: por un lado tenemos la implementación del patrón BusinessDelegate (*EJBUserFacadeDelegate*) quien leerá el nombre JNDI de la configuración obteniendo una instancia (en realidad, lo que se obtiene es un Proxy del SessionBean) para luego trabajar contra una interfaz (*UserFacade*), lo que permite usar *EJBUserFacadeDelegate* tanto desde dentro del contenedor J2EE como cliente standalone en caso de ser necesario (por ejemplo para ejecutar pruebas de unidad).

A su vez, la interfaz *UserFacade* anteriormente citada nos permitirá especificar la interfaz del SessionBean (fachada EJB del modelo) que será extendida para ser usada de manera local y remota (*LocalUserFacade* y *RemoteUserFacade* respectivamente). Finalmente, tendremos la clase *UserFacadeEJB* que proporcionará la implementación del SessionBean, implementando tanto la

fachada local como la remota. Se ha definido también la clase *UserFacadeHelper* en la que se encontrarán métodos que facilitan la conversión de entidades a objetos Transfer Object y viceversa, así como funcionalidades que son usadas en distintas acciones de la fachada (como búsquedas por identificador). Profundizando más, la clase *UserFacadeEJB* ha sido definida con la notación *@Stateful* dado que se trata de una fachada con estado (por tanto es un Stateful Session Bean). Este estado es el login del usuario, que se guarda cuando un usuario se autentica en la aplicación. Este estado será usado en varios de los métodos definidos en esta clase, algunos de los cuales se valdrán de las acciones definidas en el paquete *action* para la implementación de su funcionalidad.

Además, se incluyen una serie de excepciones dentro del subpaquete *exceptions*. Dichas excepciones son lanzadas por los métodos de esta fachada. También se incluyen los ficheros correspondientes al cifrado de la contraseña del usuario (en el subpaquete *util*).

Finalmente comentar que se incluyen objetos Custom Transfer Object que son necesarios ya que los objetos TO definidos no contienen toda la información necesaria para realizar algunas operaciones. Estos objetos se encuentran en el subpaquete *to*.

#### 4.7.4. SearchFacade

Es una fachada sin estado creada para implementar los casos de uso relativos a las búsquedas que puede realizar un usuario no autenticado en la aplicación. En cuanto a su estructura interna, se ha implementado análo-

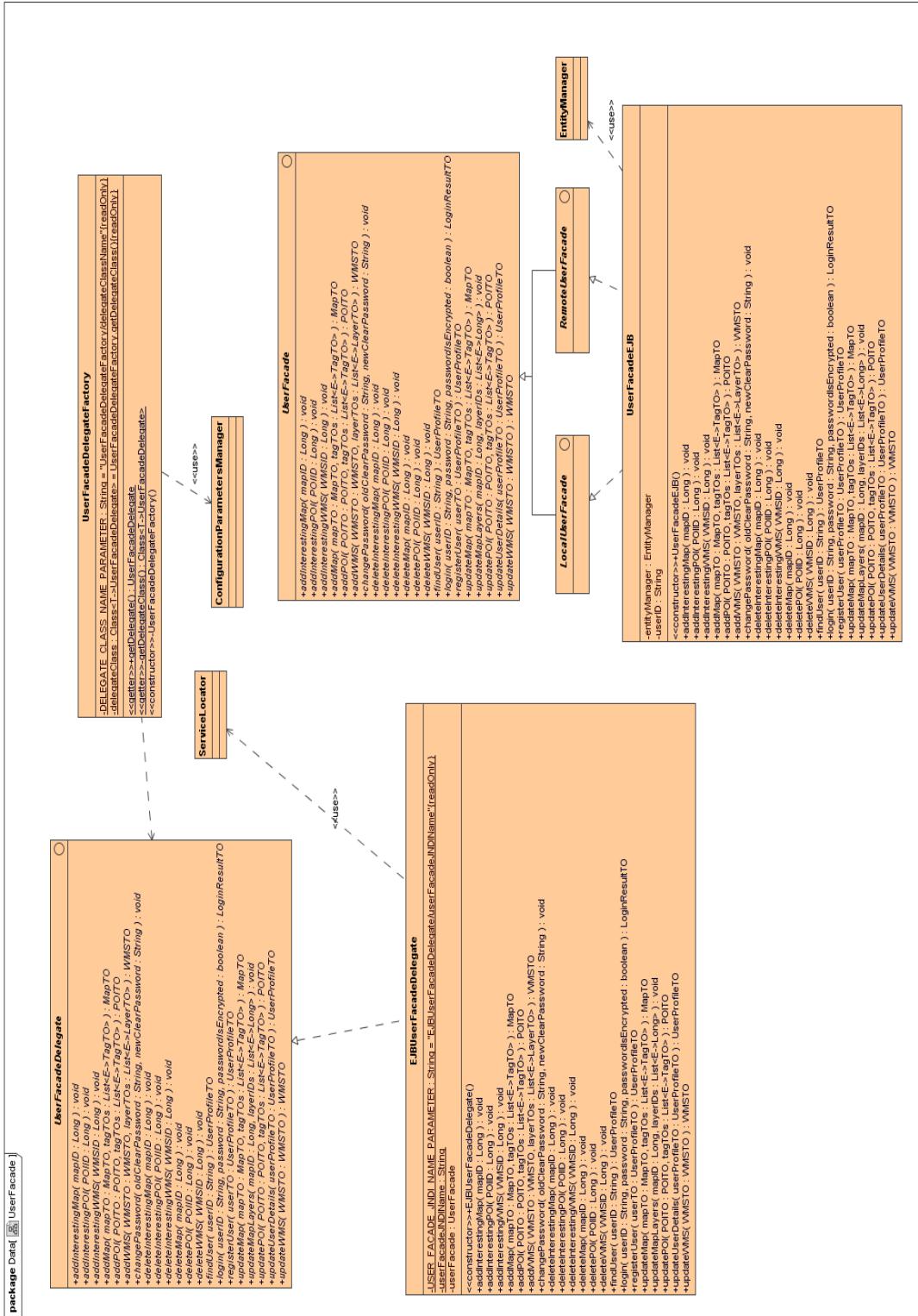


Figura 4.8: User Facade

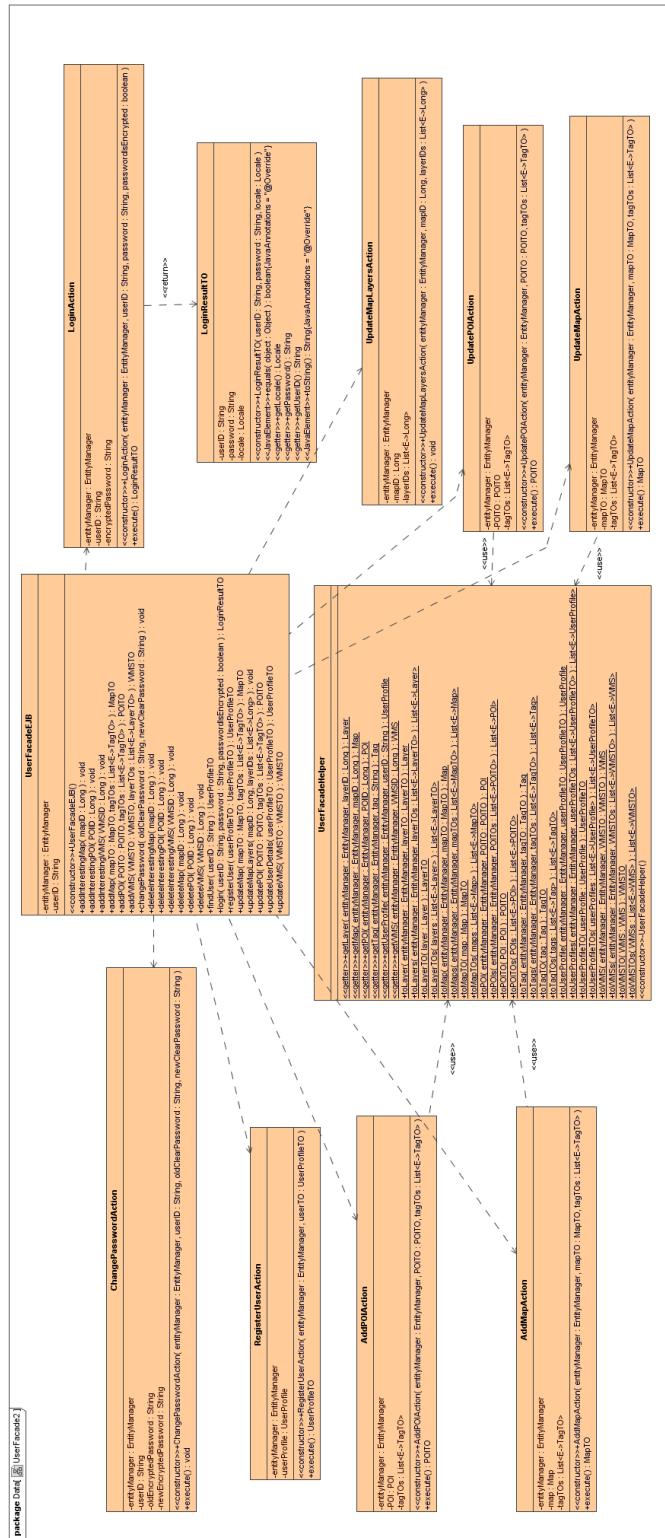


Figura 4.9: User Facade (2)

gamente a lo ya explicado en la *UserFacade*, por lo que lo único que las diferencia estructuralmente es que, como se comentó anteriormente, se trata de una fachada sin estado por lo que la clase correspondiente (*SearchFacadeEJB*) ha sido anotada con `@Stateless`, dado que se trata de un Stateless Session Bean. Además, no se tuvieron que definir excepciones propias para los métodos de esta fachada, así como tampoco fue necesaria ninguna clase utilidad como en el caso del cifrado de las contraseñas.

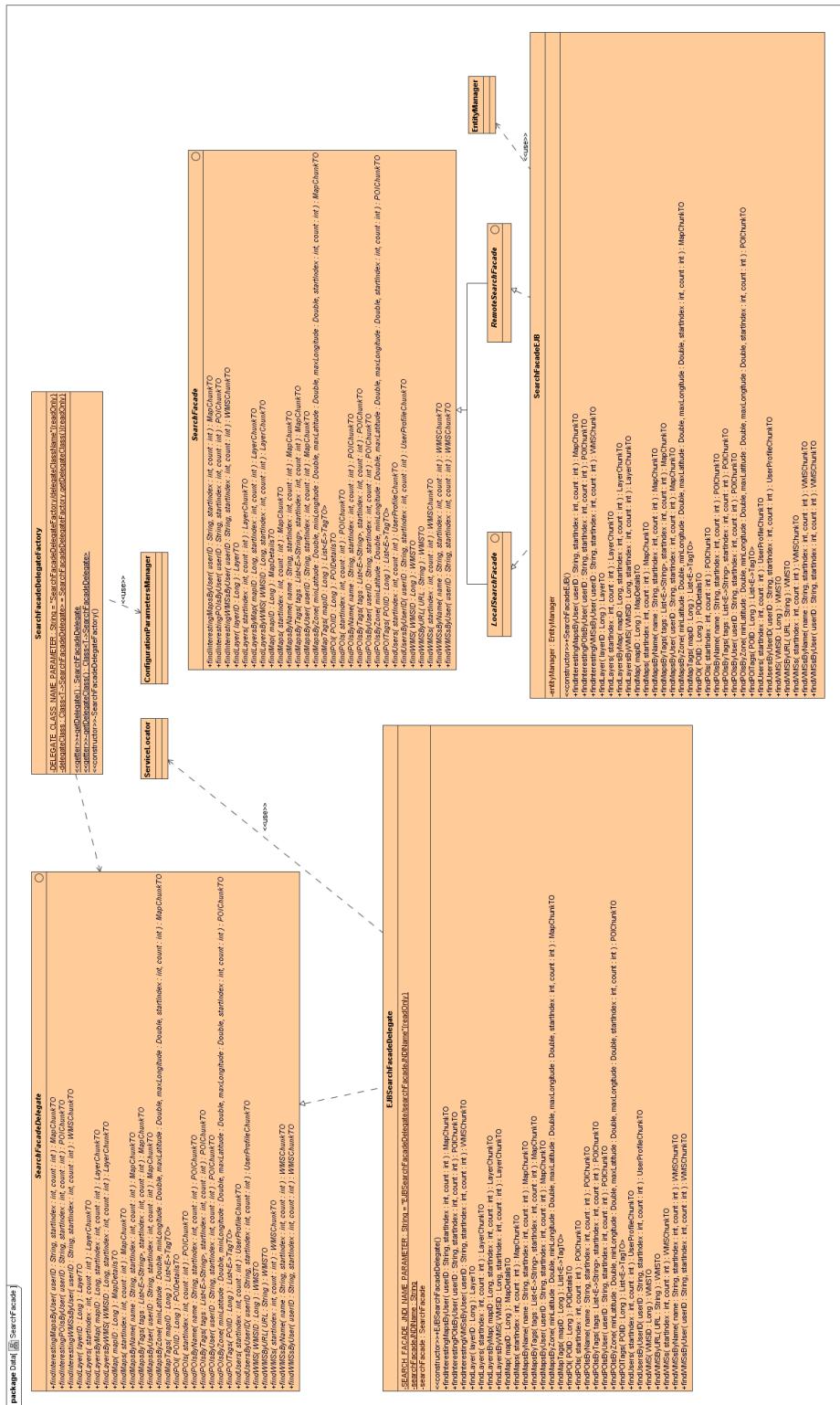


Figura 4.10: Search Facade

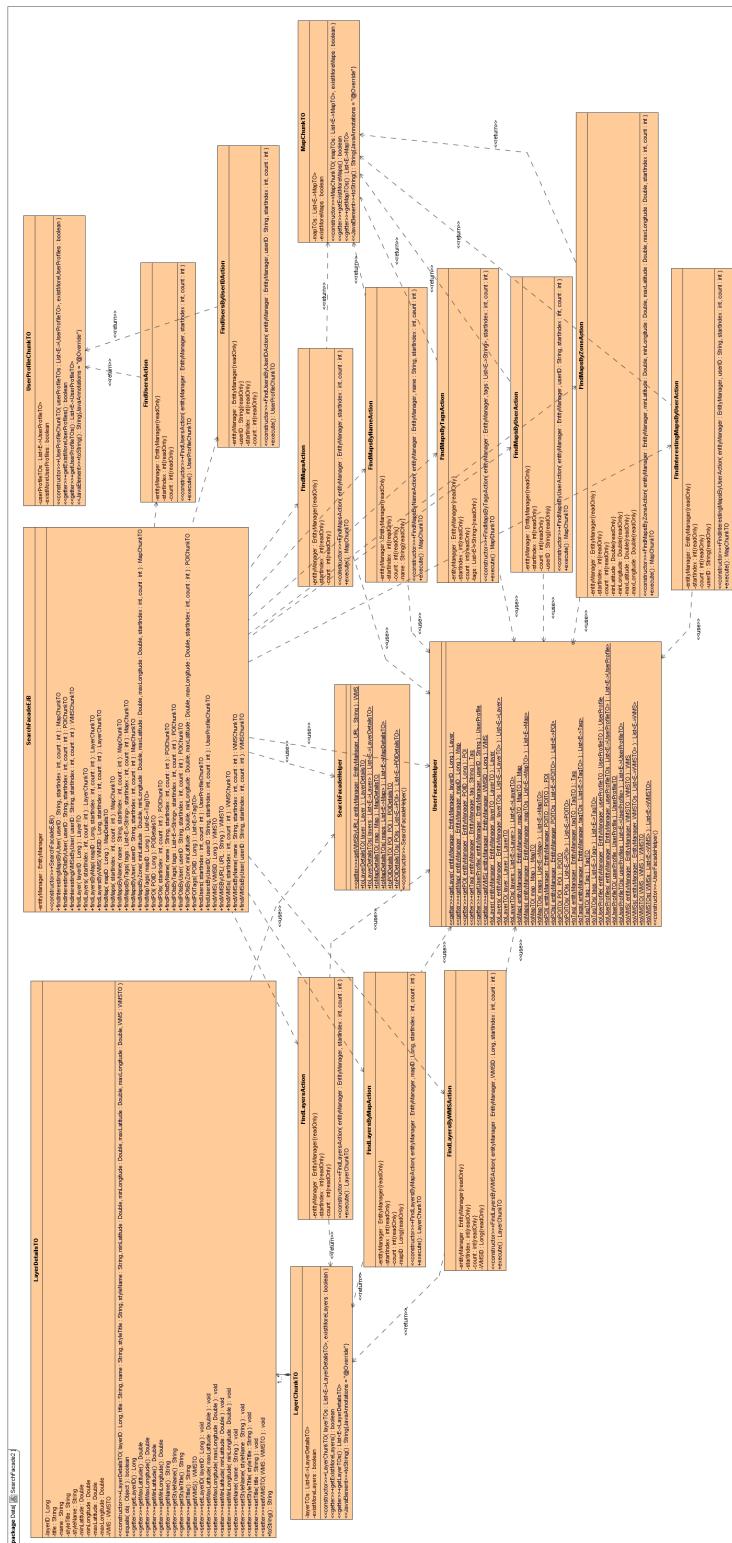


Figura 4.11: Search Facade (2)

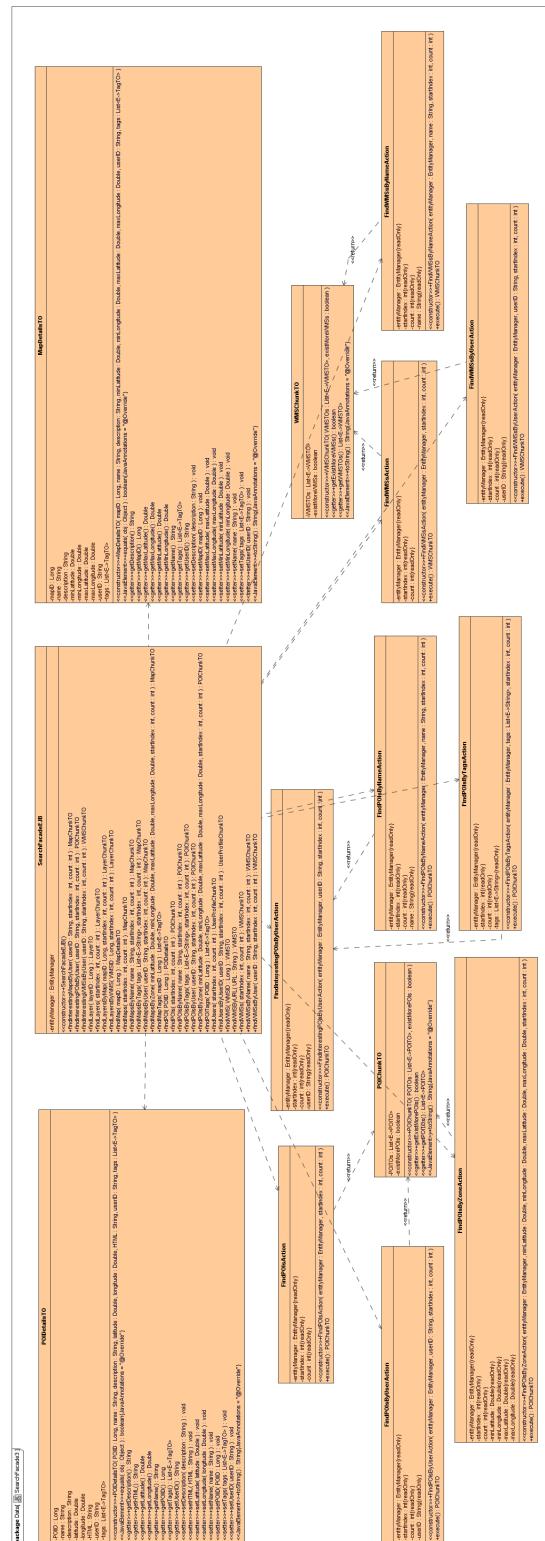


Figura 4.12: Search Facade (3)

## Patrones utilizados

En este apartado se hará una breve descripción de los patrones que han sido utilizados en este proyecto.

### 5.1. Model View Controller

Permite una separación clara entre el modelo (que constituye la lógica de negocio) y la vista (interfaz gráfica), gracias a un controlador que los mantiene desacoplados.

Las ventajas que aporta su utilización son numerosas:

- Un modelo independiente, que puede ser distribuido.
- Un modelo que puede ser reusable con distintas vistas (por ejemplo una vista web y una con interfaz de ventanas).
- Permite una separación clara del trabajo a realizar entre los miembros de un equipo de desarrollo, formado por personas con distintos grados de especialización a través de la definición de roles (así podríamos tener un diseñador web encargado de la vista y un programador para los casos de uso soportados por el modelo).

## **5.2. Layers**

Proporciona una estructuración del “software” en capas, permitiendo la ocultación de las tecnologías empleadas por el sistema y favoreciendo la división clara de trabajo entre los miembros de un equipo.

Suele combinarse con el anterior patrón mencionado para lograr, por ejemplo, que tanto la vista como el controlador nunca conozcan las tecnologías usadas en la implementación del modelo.

## **5.3. Tiles**

Proporcionado por Struts, puede ser considerado como una evolución del antiguo sistema de plantillas Template y un caso particular del patrón Composite View. Cada pantalla es la composición de varias páginas JSP de modo que podemos clasificar las pantallas en grupo según su estructura aislando las partes comunes en páginas JSP.

Se deberá definir cada pantalla en un fichero XML de manera que:

- Por cada grupo se defina una pantalla base, que especifica la página JSP que contiene la estructura y da valor a las partes comunes.
- Cada pantalla de un grupo se defina como extensión de la pantalla base, dando valor a las partes específicas (e incluso sobreescribiendo alguna heredada).

#### 5.4. Transfer Object

Este patrón pretende agrupar un conjunto de atributos procedentes de uno o varios objetos del dominio. En cuanto a la estructura general del patrón es la de implementar la interfaz *java.io.Serializable* y poseer métodos *get* para el acceso a los atributos y métodos *set* para aquellos atributos que sean modificables.

#### 5.5. Business Delegate

El Business Delegate se utiliza para reducir el acoplamiento entre los clientes de la capa de presentación y los servicios de negocio. Oculta tanto los detalles de la implementación del servicio de negocio, como los detalles de búsqueda y acceso de la arquitectura EJB.

El Business Delegate actúa como una abstracción de negocio del lado del cliente y, por lo tanto, oculta la implementación de los servicios del negocio. Mediante su utilización se reduce el acoplamiento entre los clientes de la capa de presentación y los servicios de negocio del sistema.

Dependiendo de la estrategia de implementación, podría aislar a los clientes de la posible volatilidad en la implementación del API de los servicios de negocio. Potencialmente, esto reduce el número de cambios que se deben hacer en el código del cliente de la capa de presentación cuando cambie el API del servicio de negocio o su implementación subyacente.

Cuando el Business Delegate se utiliza con un Session Facade, normalmente hay una relación uno a uno entre los dos. Esta relación existe porque la lógica,

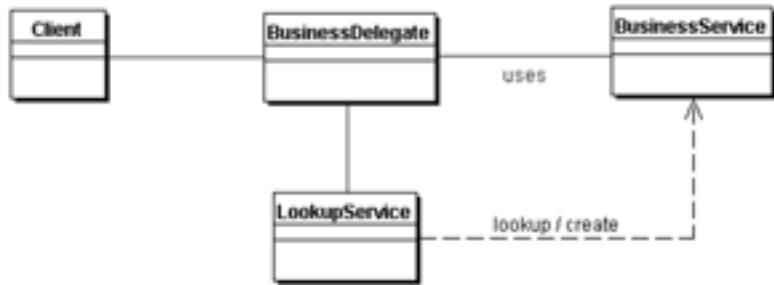


Figura 5.1: Patrón Business Delegate

que podría haber sido encapsulada en un Business Delegate en relación a su interacción con varios servicios de negocio (creando una relación uno a uno), normalmente se construye de vuelta en un Session Facade.

## 5.6. Page-by-Page Iterator

Permite acceder a una lista grande de Transfer Objects de una manera eficiente. Es especialmente útil cuando el usuario está interesado en visualizar la lista de Transfer Objects en trozos, pudiendo ir hacia adelante o atrás, cuando la lista completa no cabría en la pantalla o bien cuando la lista completa no cabría en memoria.

## 5.7. Service Locator

Permite abstraer la lógica de localización/creación de servicios. Comúnmente un Business Delegate que proxifica a un Session Bean utiliza este patrón para localizar/crear la instancia del Session Bean al que proxifica.

## 5.8. Facade

Proporciona una interfaz unificada para un conjunto de interfaces de un subsistema, lo que evita la comunicación y dependencias entre las mismas haciendo que se comuniquen entre sí únicamente a través de sus fachadas. Su aplicación se lleva a cabo cuando se quiere dotar a un subsistema complejo de una única interfaz de más alto nivel, de un único punto de entrada en el que aparece un método por cada caso de uso implementado. Mediante su aplicación se evita el acoplamiento entre el cliente y las clases del sistema.

## 5.9. Factory method

Permite definir una interfaz para crear un objeto, pero deja que sean las subclases quienes decidan qué clase instanciar. Su uso es de utilidad cuando:

- Una clase no puede prever la clase de objetos que debe crear.
- Una clase quiere que sean sus subclases quienes especifiquen los objetos que ésta crea.
- Las clases delegan la responsabilidad en una de entre varias clases auxiliares, y queremos localizar qué subclase auxiliar concreta es en la que se delega.

En el proyecto se usa principalmente para obtener una instancia de cada una de las fachadas implementadas.

## **5.10. Composite**

Patrón estructural que tiene como objetivo permitir componer objetos en estructuras árbol para representar jerarquías parte-todo para, de esta forma, permitir a los clientes tratar de forma uniforme a los objetos y a las composiciones.

Existen variedad de implementaciones pero la utilizada en el proyecto es la de mantener referencias de los componentes hijos a sus padres, para intentar simplificar el recorrido y la gestión de la estructura compuesta, manteniendo el invariante de que todos los hijos de un compuesto tienen como parente al compuesto que a su vez los contiene a todos ellos como hijos.

## **5.11. Singleton**

También conocido como instancia única. Asegura que una clase sólo tiene una instancia y proporciona un punto global de acceso a ella. Debe existir una única instancia de una clase, y ésta debe ser accesible a los clientes desde un punto de acceso bien conocido. La única instancia puede ser extendida sin modificar su código.

## **5.12. Chain of Responsibility**

El objetivo de este patrón de comportamiento es el de evitar acoplar el emisor de una petición a un receptor, dando a más de un objeto la posibilidad de responder a la petición. Para lograrlo, encadena los objetos receptores y pasa la petición a través de una cadena hasta que es procesada por algún

objeto. Utilizado en este proyecto en la cadena de filtros del *front controller* del controlador de la aplicación web.

### 5.13. Front Controller

Este patrón proporciona un punto de entrada centralizado que controla y maneja las peticiones del Cliente, que están relacionadas con el procesamiento de negocio y el control de flujo. El ServletFront proporciona una implementación del controlador como un servlet. Claramente esta visión está orientada a Web, puesto que un servlet es una clase Java que puede recibir peticiones, generalmente HTTP, generando una salida.

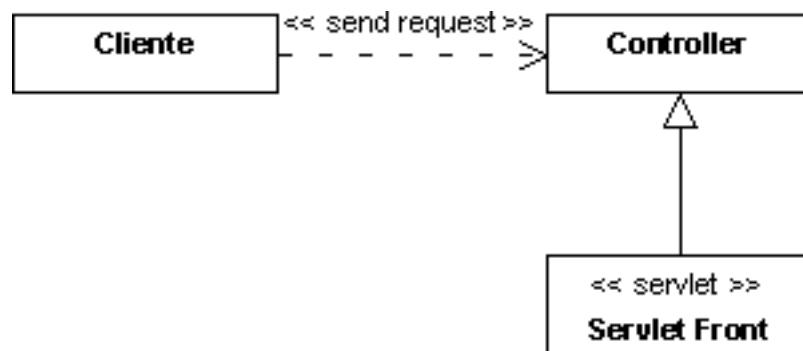


Figura 5.2: Patrón Front Controller



# 6

## Manual de usuario

En este apartado se mostrarán las distintas acciones que puede realizar un usuario por medio de una serie de pantallas que facilitarán la explicación. Al acceder a la aplicación por medio de un navegador web nos aparece la pantalla principal de la misma.

Podemos observar una parte superior o encabezado, en el que aparece el nombre de la aplicación y más abajo un mensaje de bienvenida a la aplicación y una serie de botones por los que podremos acceder a las distintos grupos de acciones disponibles.

En la parte inferior de la página vemos un pie de página en el que se muestra información de contacto del creador de la aplicación por si se quieren hacer sugerencias para mejorar dicha aplicación.

Por último, en la parte central se muestra por una parte las distintas opciones del grupo de acciones seleccionado en este momento y a la derecha el contenido en si de la página. Por defecto, si no hay contenido que mostrar se muestra el logotipo de la aplicación.

Para poder sacarle el máximo partido a la aplicación vamos a crear un usuario. Para ello pulsamos sobre el botón “Entrar”.

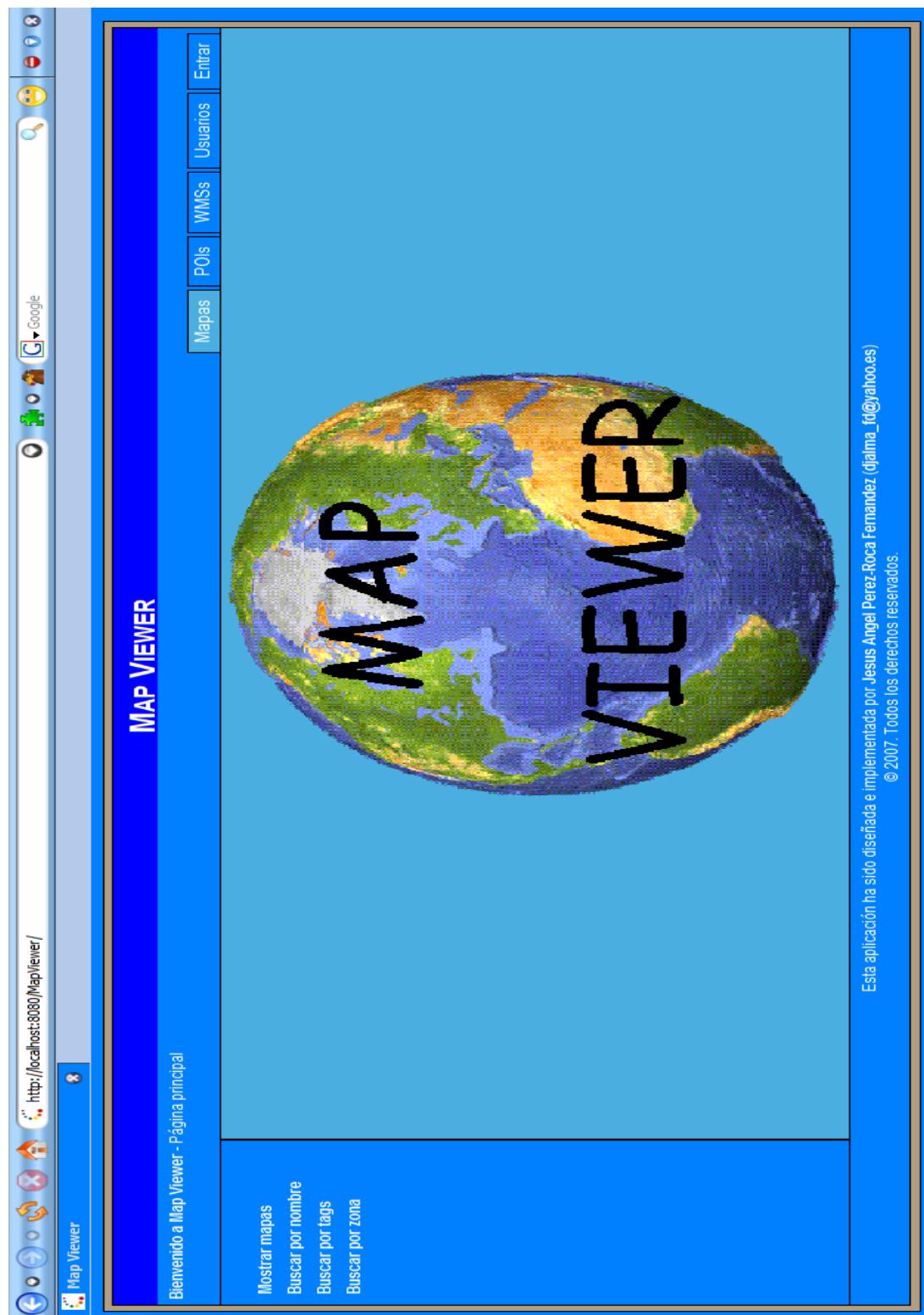


Figura 6.1: Pantalla principal usuario no autenticado

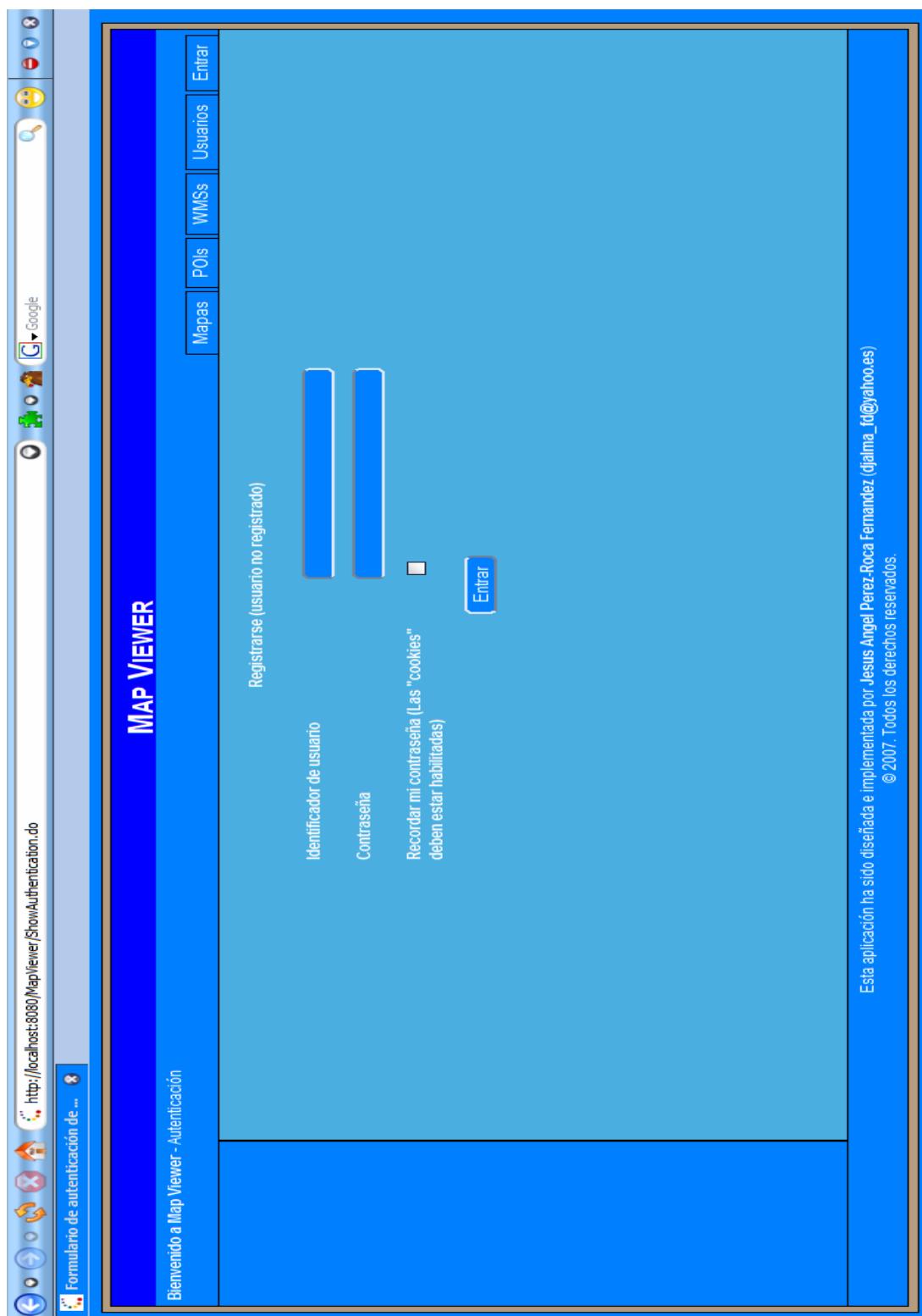


Figura 6.2: Pantalla de autenticación

En esta pantalla podríamos autenticarnos en la aplicación si tuviésemos un usuario creado, introduciendo para ello el identificador de usuario (login) y contraseña. También podríamos marcar la opción de que el navegador recuerde nuestros datos si dejamos la sesión abierta. Estos datos se guardan durante un máximo de 30 días, siempre y cuando las cookies estén habilitadas y la sesión no se cierre.

Como no tenemos aún ningún usuario creado, pulsamos sobre el enlace para registrarnos.

En esta pantalla introduciremos nuestros datos personales, así como nuestro identificador de usuario deseado (login) y nuestra contraseña (mínimo 6 caracteres) por duplicado para evitar errores de escritura. Tras introducir todos los datos y pulsar el botón “Aceptar” volvemos a la pantalla de inicio de la aplicación, pero ya como usuario autenticado.

Observamos que el mensaje genérico de bienvenida a la aplicación ha sido sustituido por otro personalizado con nuestro identificador de usuario (login) y observamos también cambios en la barra de botones: el botón “Entrar” ha sido sustituido por otro que nos permite salir de la aplicación (“Salir”) y apareció un nuevo botón a la izquierda de la barra de botones que nos permite modificar nuestros datos personales (“Actualizar perfil”). Si pulsamos sobre dicho botón nos aparecerá una pantalla parecida a la que aparecía al registrarse.

Aquí podríamos hacer cambios en los datos y, tras darle al botón “Aceptar”, dichos cambios quedarían registrados en la base de datos. Si quisieramos cambiar nuestra contraseña, deberíamos pulsar sobre el enlace que aparece.

The screenshot shows a web application window titled "MAP VIEWER". At the top, there is a navigation bar with tabs: "Mapas", "POIs", "WMSS", "Usuarios", and "Entrar". Below the navigation bar, there is a registration form with the following fields:

- Identificador de usuario:
- Contraseña:
- Repetir la contraseña:
- Nombre:
- Apellidos:
- E-mail:
- Idioma:

At the bottom right of the form is a blue "Aceptar" button.

On the left side of the application, there is a sidebar with the following links:

- Mostrar usuarios
- Buscar por identificador de usuario
- Formulario de registro de usuario

At the very bottom of the application window, there is a footer with the following text:

Esta aplicación ha sido diseñada e implementada por Jesus Angel Perez-Roca Fernandez (djalmalfd@yahoo.es)  
© 2007. Todos los derechos reservados.

Figura 6.3: Pantalla de registro



Figura 6.4: Pantalla principal usuario autenticado

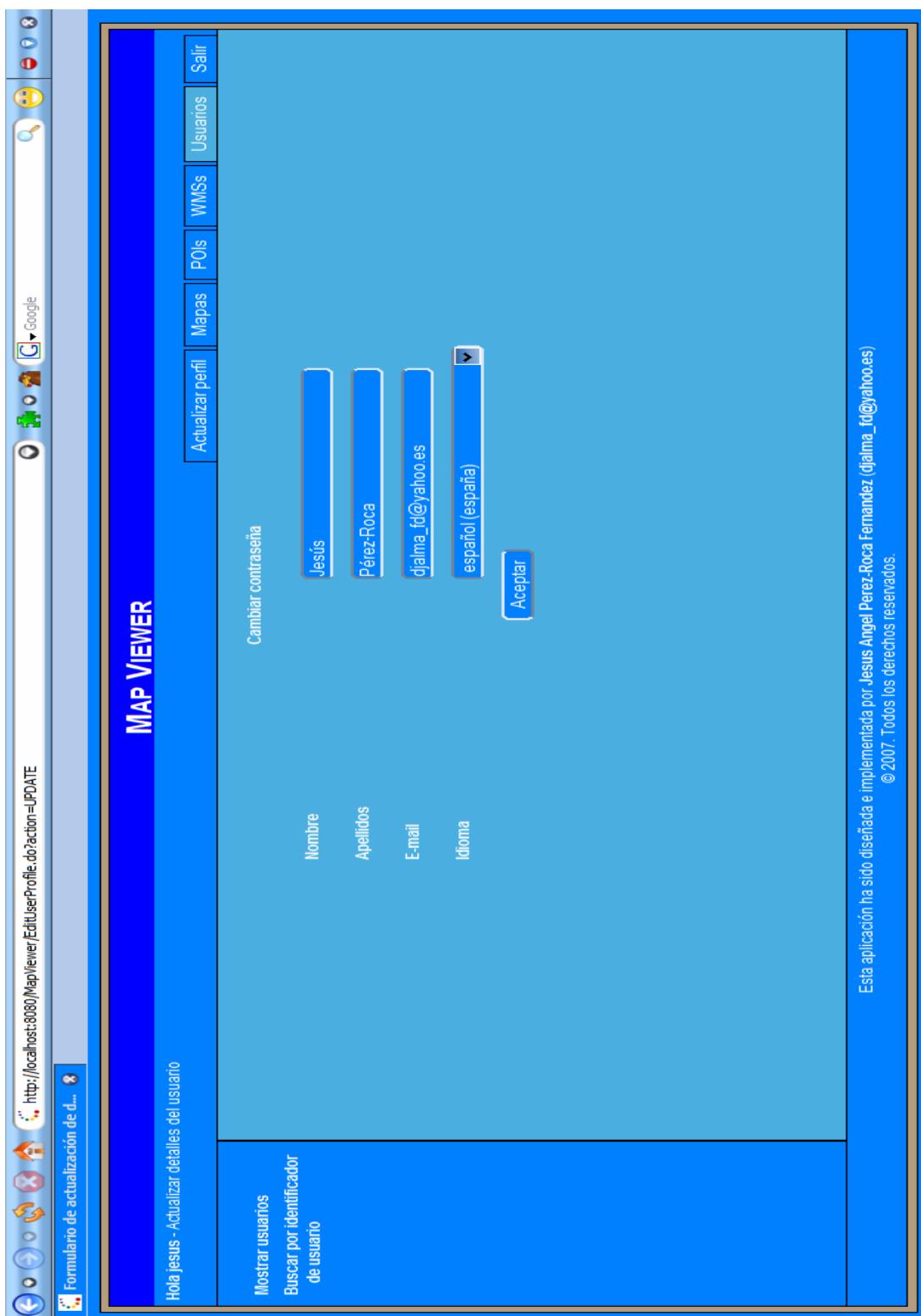


Figura 6.5: Pantalla de actualización de perfil

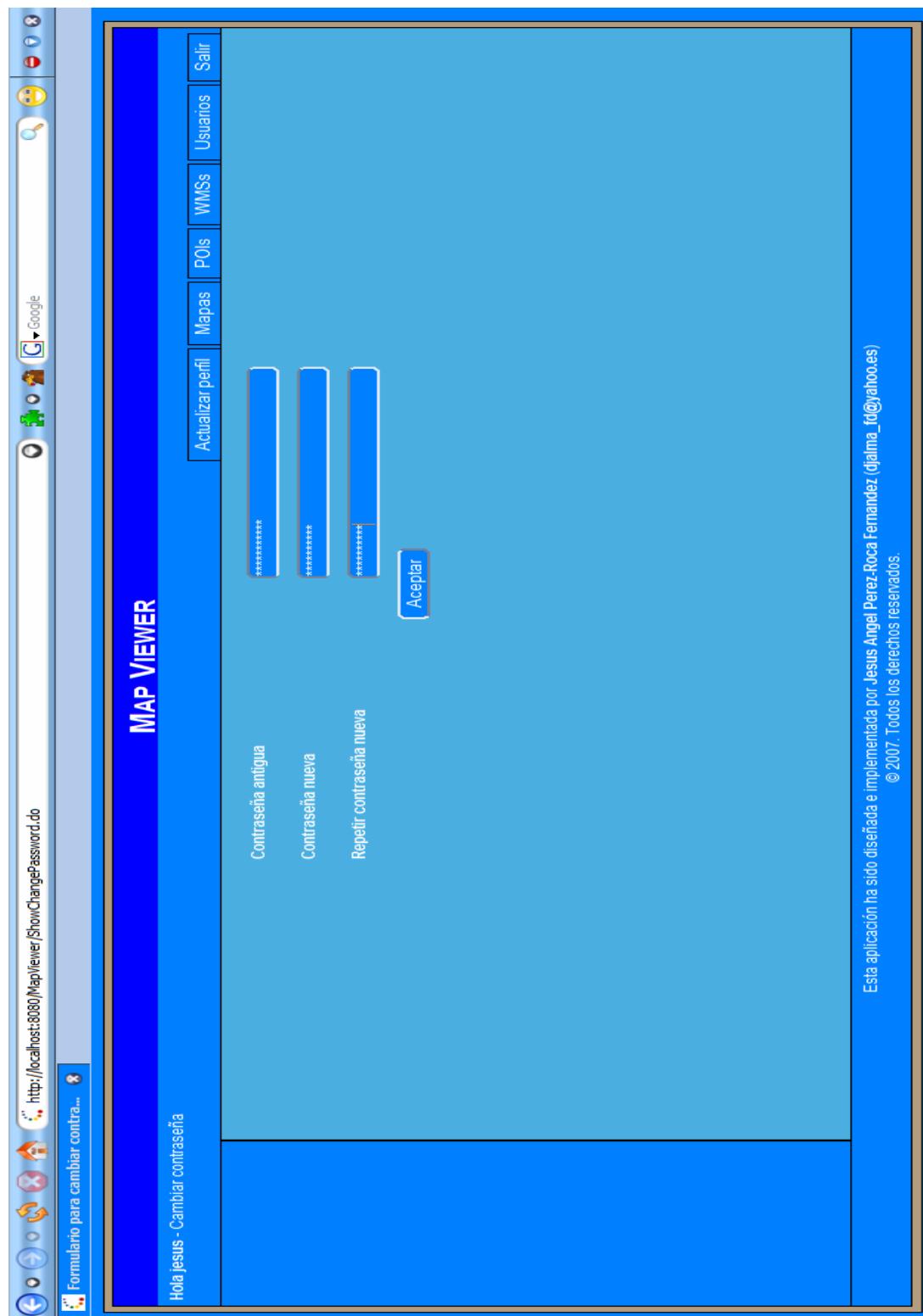


Figura 6.6: Pantalla de cambio de contraseña

Aquí deberemos introducir tanto la contraseña antigua como la nueva por duplicado, para evitar por una parte que alguien que se haya encontrado con nuestra sesión abierta pueda cambiar nuestra contraseña y por otro lado para evitar que cometamos un error.

Una vez estamos conformes con nuestros datos podremos proceder a crear nuestro primer mapa. Para ello, primero deberíamos crear primero un servidor WMS que nos va a proporcionar las capas que se mostrarán en el mapa. Pulsamos primero sobre el botón “WMSs” de la barra de botones. Notamos como las opciones de la barra lateral cambian para ofrecernos las opciones relacionadas con los servidores WMS.

Antes de crear nuestro WMS podríamos comprobar que no existe un WMS que nos pueda valer para nuestro mapa. Para ello existen tres formas de búsqueda:

- “Mostrar WMSs”, que nos muestra todos los WMSs existentes en la base de datos (en grupos de 5 con enlaces “Siguiente” y “Anterior”, en caso de ser necesarios).
- “Buscar por nombre”, que nos permite filtrar los WMSs existentes por medio de un cuadro de búsqueda, en el que introduciremos el nombre a buscar. También nos muestra los WMSs en grupos de 5 con enlaces “Siguiente” y “Anterior”, en caso de ser necesarios.
- “Mis WMSs”, que nos muestra los WMSs que hemos creado.
- “WMSs interesantes”, que nos muestra los WMSs que hemos añadido a nuestra lista de favoritos.

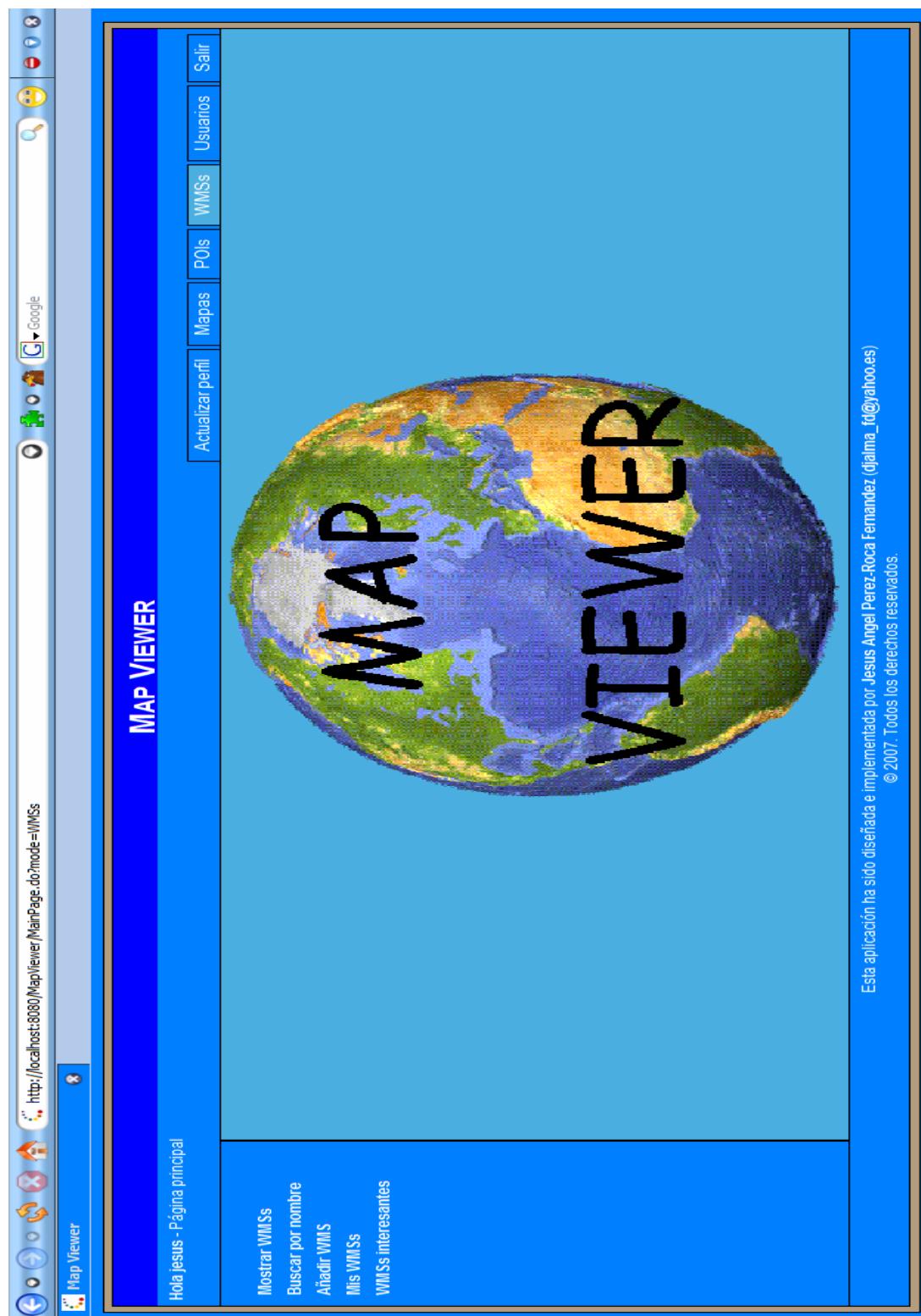


Figura 6.7: Pantalla principal (opciones WMSS)

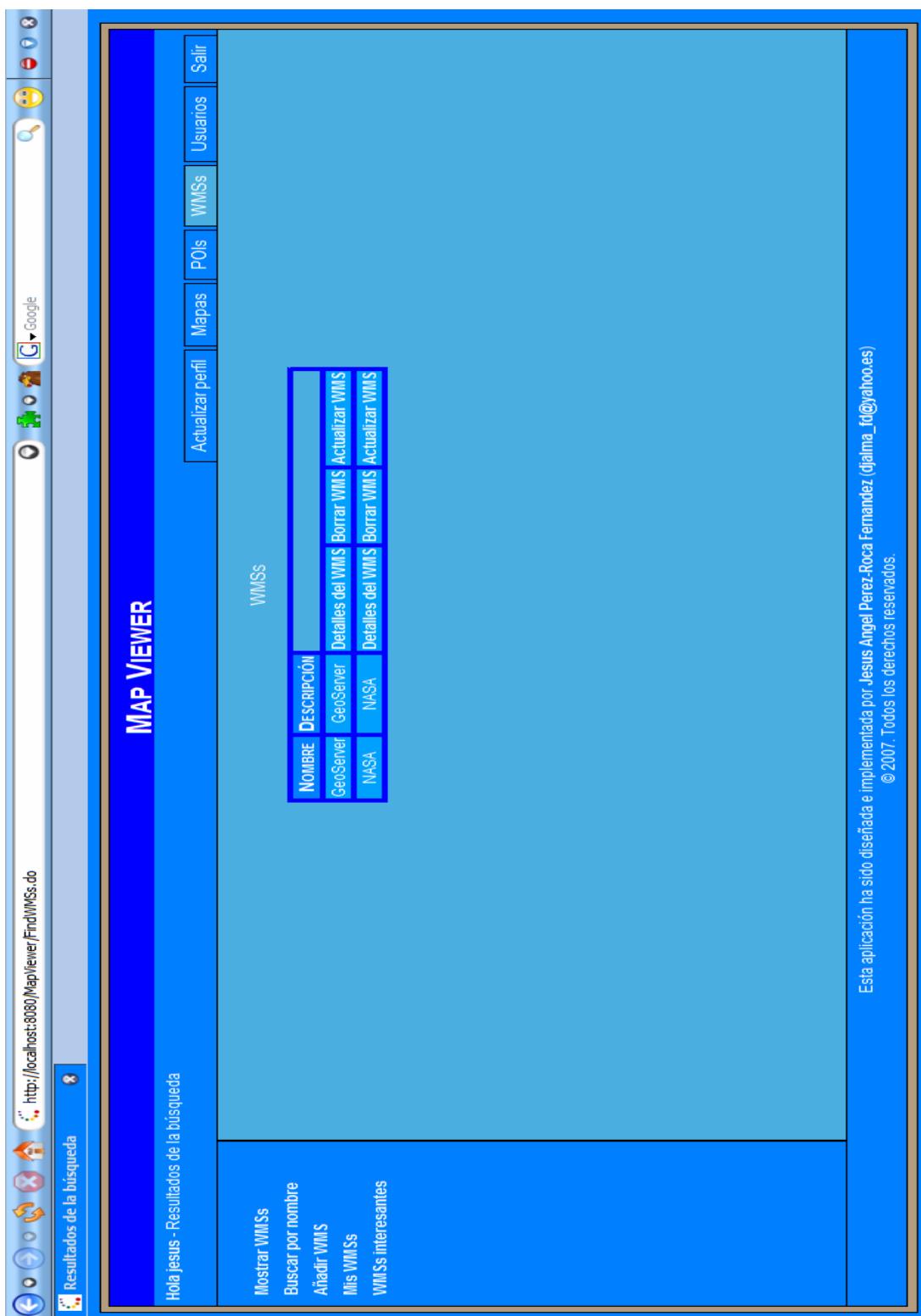


Figura 6.8: Búsqueda de WMSs

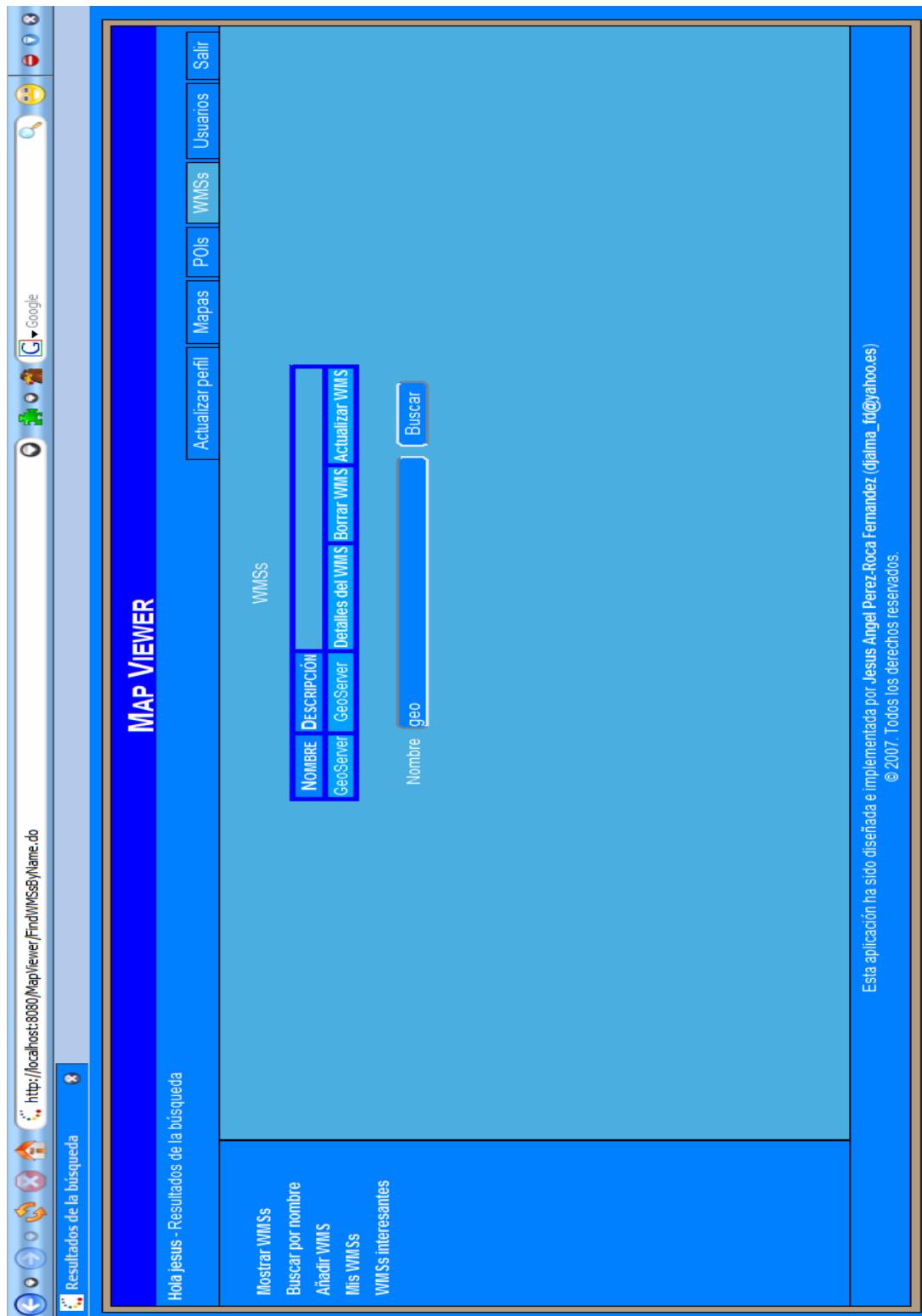


Figura 6.9: Búsqueda de WMSS por nombre

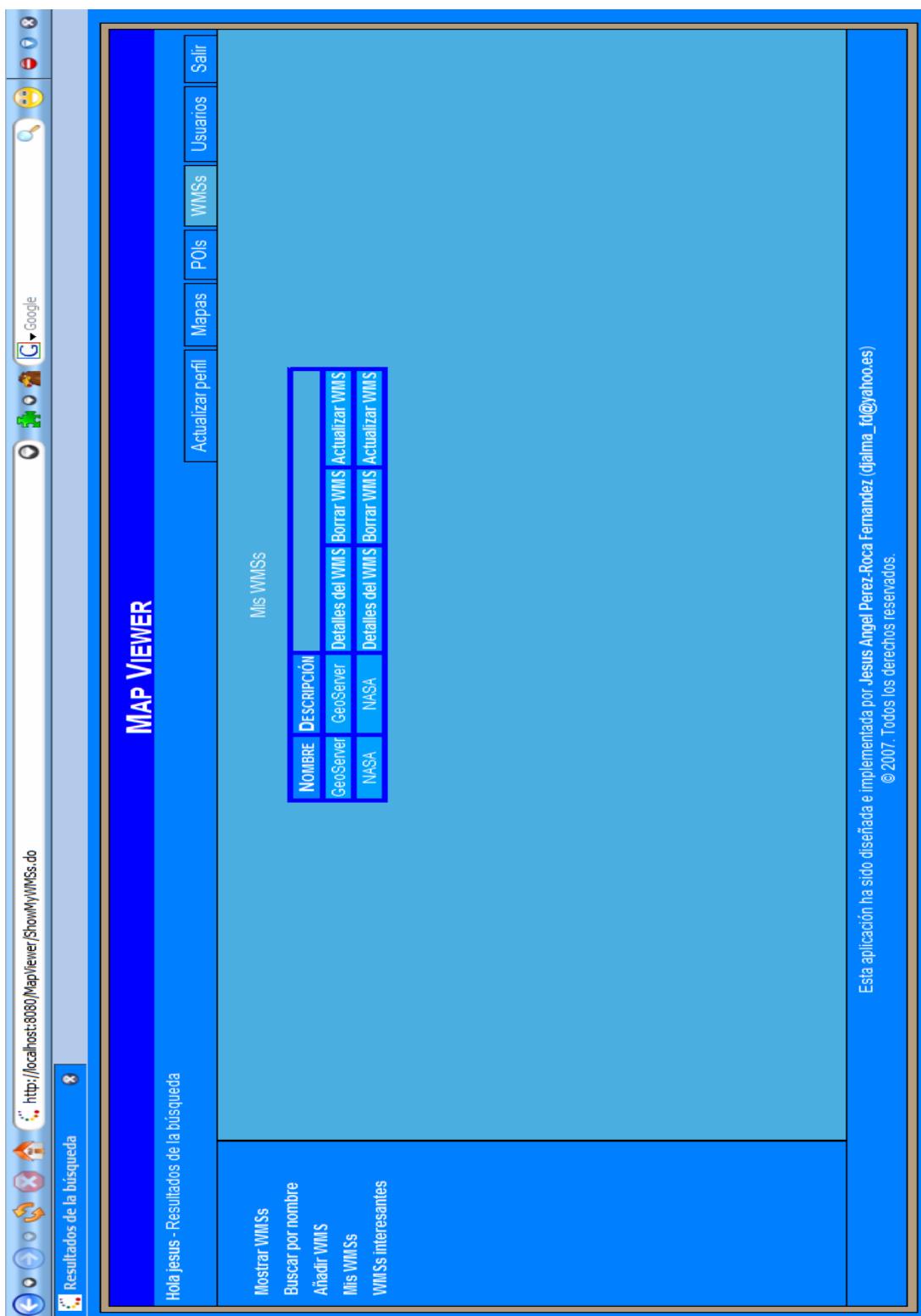


Figura 6.10: Pantalla de “Mis WMSs”

La información obtenida con cualquiera de los métodos es análoga y nos mostrará una serie de enlaces adicionales, dependiendo de si el WMS lo hemos creado nosotros o no y de si lo tenemos en nuestra lista de WMSs preferidos o no: si somos los creadores del WMS nos dejará actualizarlo o borrarlo (en este caso nos devuelve a la página principal) y si no lo tenemos en nuestra lista de WMSs preferidos nos dejará añadirlo (y nos dejará quitarlo si ya lo tenemos en la lista), tras lo cual nos devuelve a la página principal.

En cualquier caso, siempre nos mostrará un enlace en el que podremos ver más detalladamente el WMS en cuestión.

Supongamos que el WMS que necesitamos no está en la base de datos. En ese caso, deberíamos pulsar sobre el enlace de “Añadir WMS”.

En esta pantalla introduciremos un nombre y una descripción para el WMS, así como su URL. Ésta debe terminar en ?. Tras introducir los datos pulsaremos sobre el botón “Aceptar” y regresaremos a la pantalla principal. El siguiente paso sería crear el mapa. Para ello pulsamos primero sobre el botón “Mapas” de la barra de botones para que nos aparezcan las opciones relativas a los mapas y luego pulsamos el enlace “Añadir mapa”.

En esta pantalla deberemos introducir un nombre y una descripción para el mapa. También deberemos introducir una serie de etiquetas (tags) para facilitar la posterior búsqueda del mapa. Por último deberemos especificar las coordenadas del mapa. Para facilitar esta acción existe un selector de zona geográfica que nos permite dibujar una superficie rectangular sobre un mapa del mundo que será la zona en la que estará definido el mapa. Este “popup” se puede mover por si es demasiado grande y no cabe en la pantalla

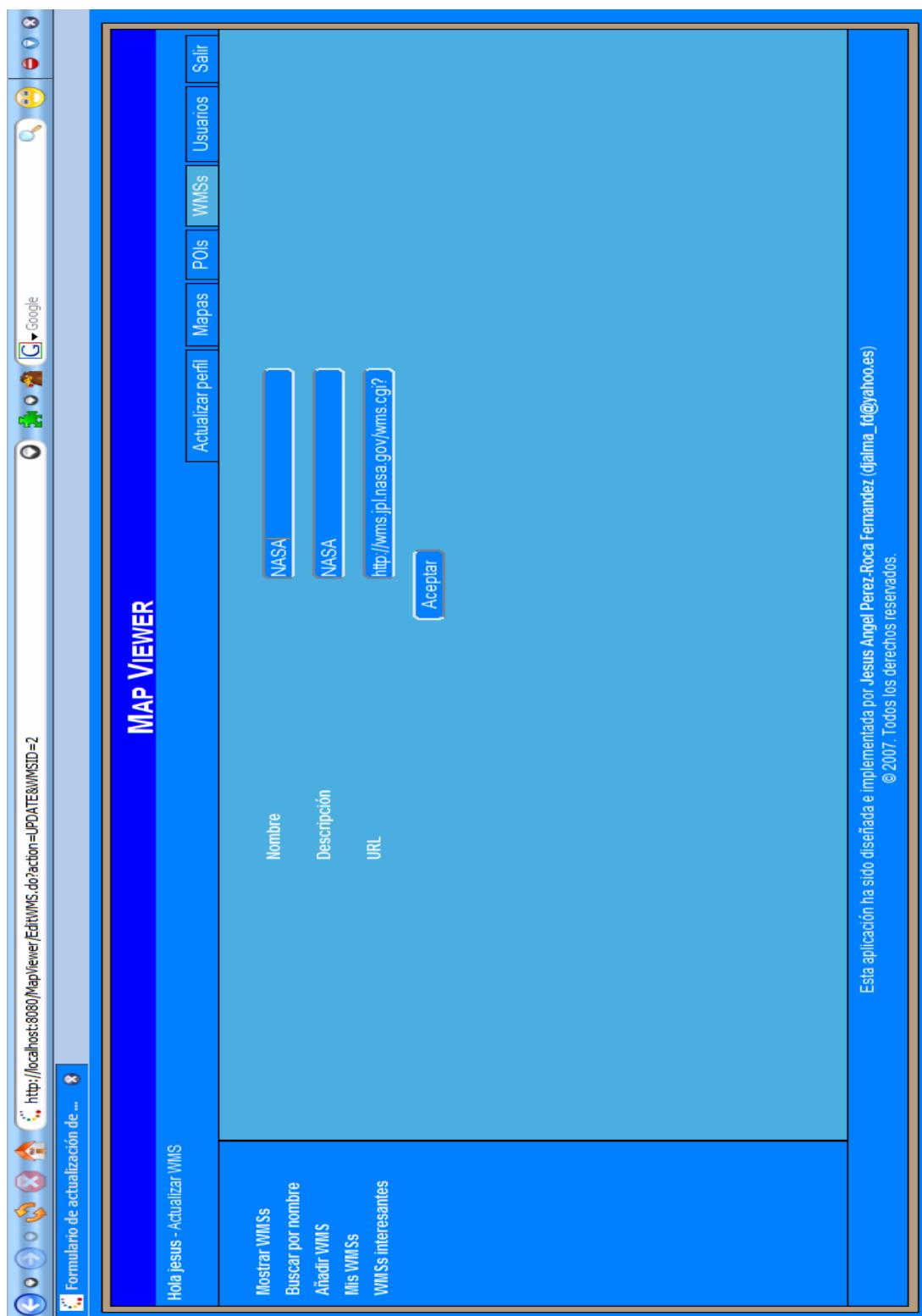


Figura 6.11: Pantalla de actualización de WMS

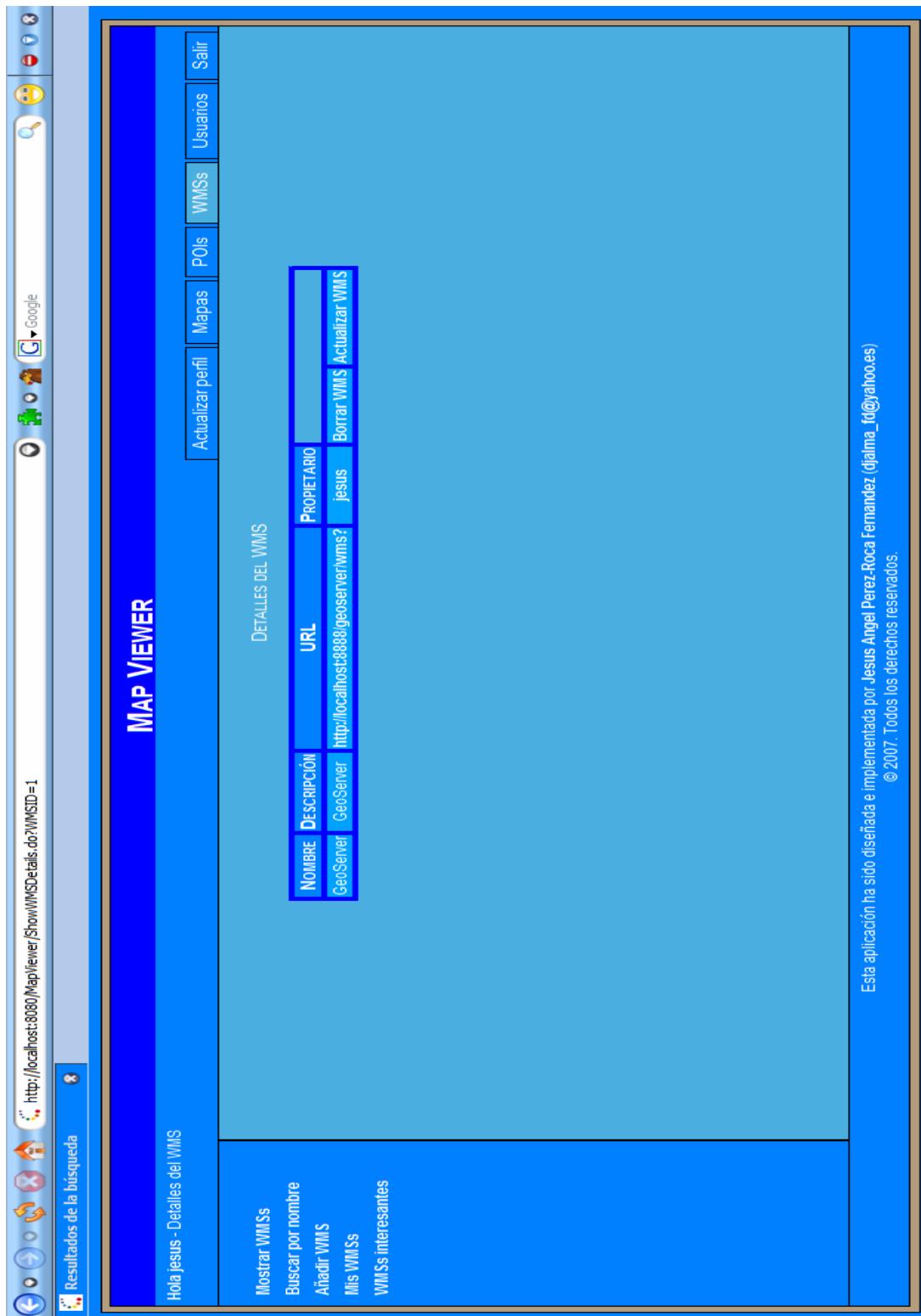


Figura 6.12: Pantalla de detalles de WMS

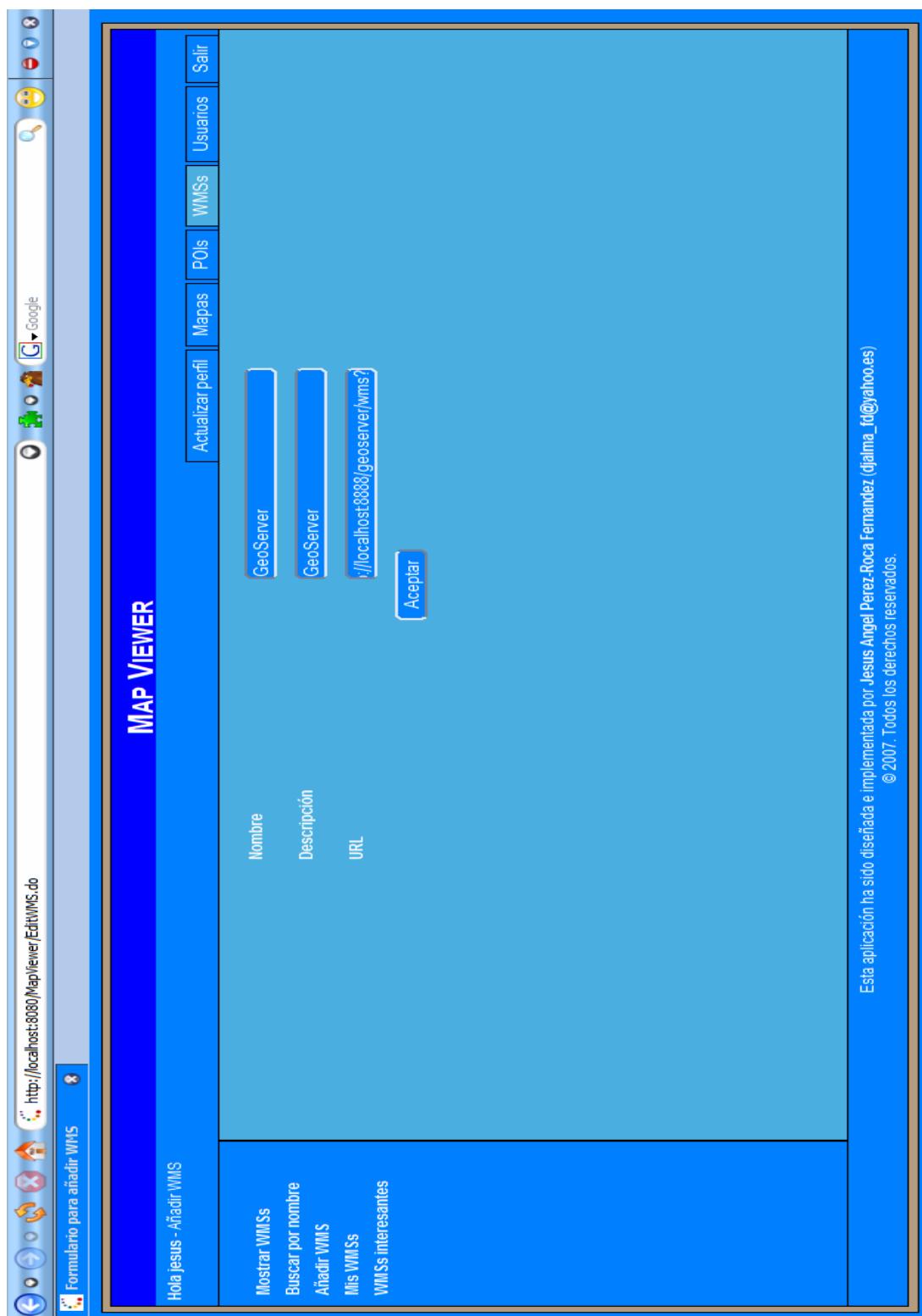


Figura 6.13: Pantalla de añadir WMS

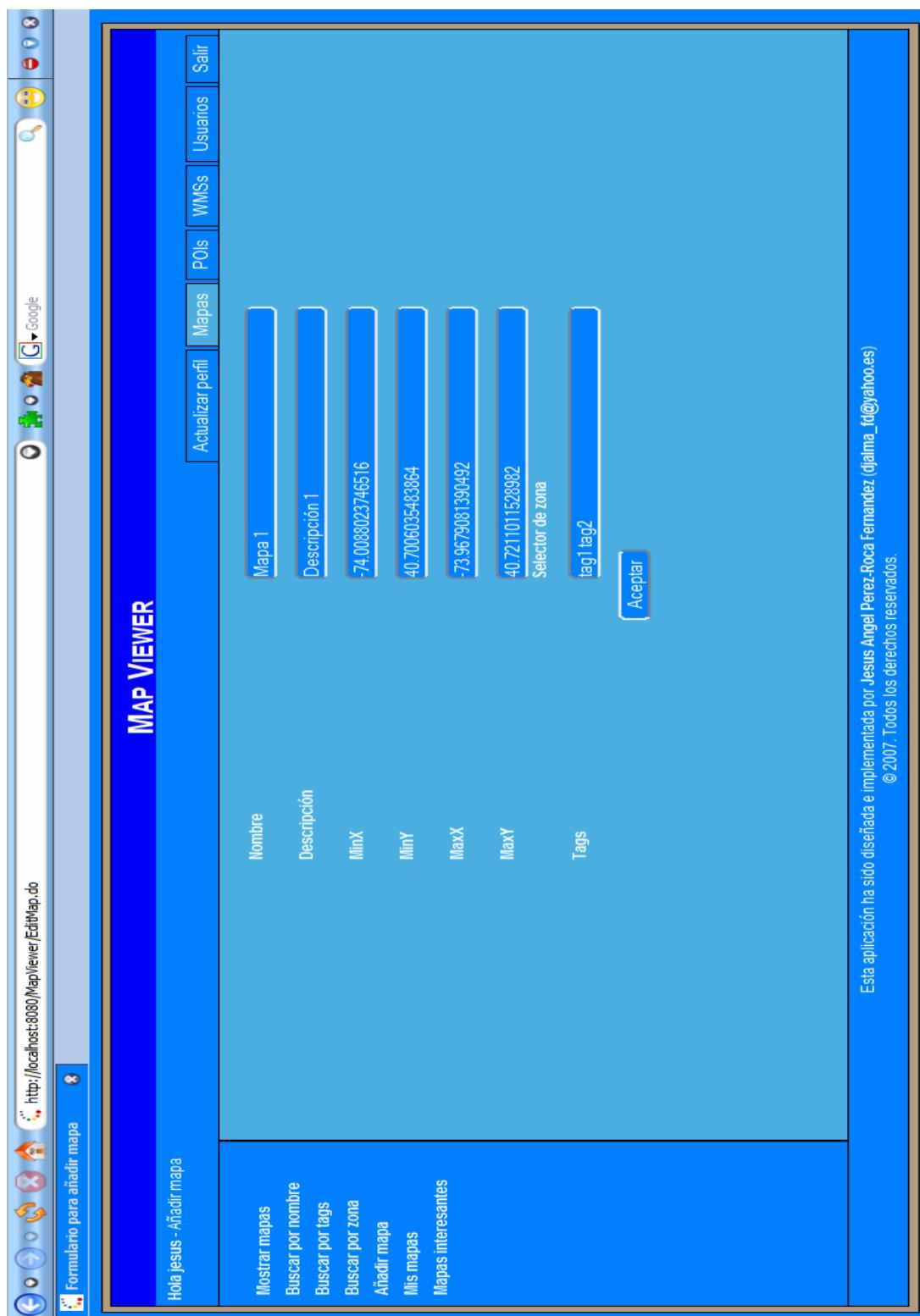


Figura 6.14: Pantalla de añadir mapa

del navegador o por si nos tapa una parte de la pantalla que queremos ver.

Tras darle al botón “Aceptar” en el selector de zona nos aparecen introducidos los valores seleccionados en su lugar correspondiente. Si le damos al botón “Aceptar” de la pantalla de “Añadir mapa” pasaremos a la siguiente pantalla.

En esta pantalla nos aparecerán en un desplegable todos nuestros WMSs, así como los WMSs que tengamos en nuestra lista de WMSs preferidos, para que podamos escoger de qué WMS vamos a seleccionar las capas. Una vez escogido el deseado le daremos al botón “Aceptar” para pasar a la pantalla de selección de capas.

Aquí veremos una lista con todas las capas y estilos de ese WMS disponibles en la base de datos. Por medio de los enlaces “Añadir capa” y “Eliminar capa” podremos ir escogiendo las capas que va a tener el mapa, sabiendo en todo momento cuales ya están añadidas para no seleccionarlas por duplicado. Como normalmente serán bastantes capas, éstas se muestran en grupos de 5 con enlaces “Siguiente” y “Anterior”, en caso de ser necesarios. Una vez hayamos escogido todas las capas que queramos que tenga el mapa, pulsaremos el botón “Aceptar”, momento en el cual todos los datos serán transferidos a la base de datos con lo que ya tendremos nuestro mapa creado.

Ahora podríamos pasar a visualizarlo. Para ello lo podemos buscar por distintos métodos:

- “Mostrar mapas”, que nos muestra todos los mapas existentes en la base de datos (en grupos de 5 con enlaces “Siguiente” y “Anterior”, en caso de ser necesarios).

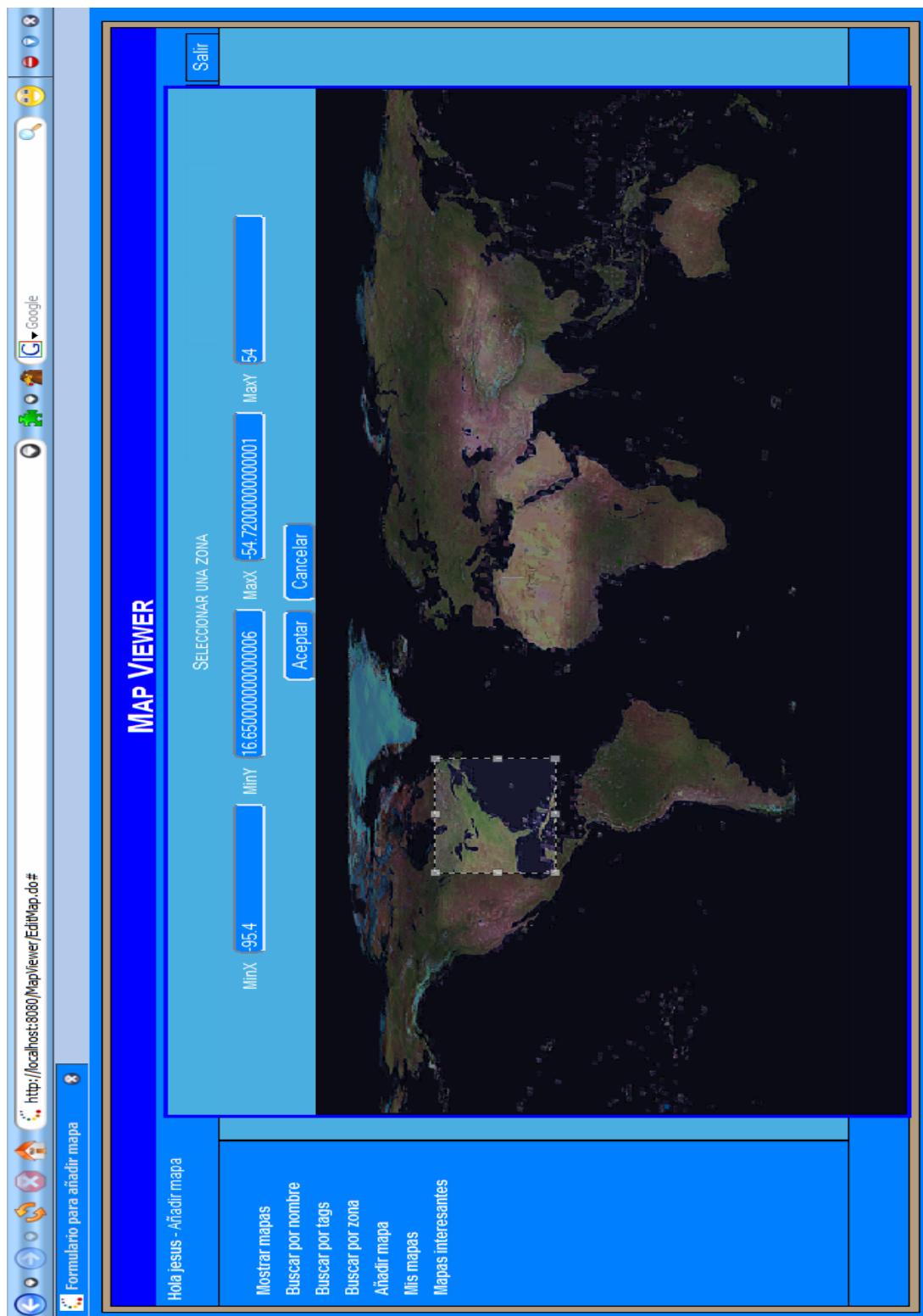


Figura 6.15: Pantalla selector zona geográfica

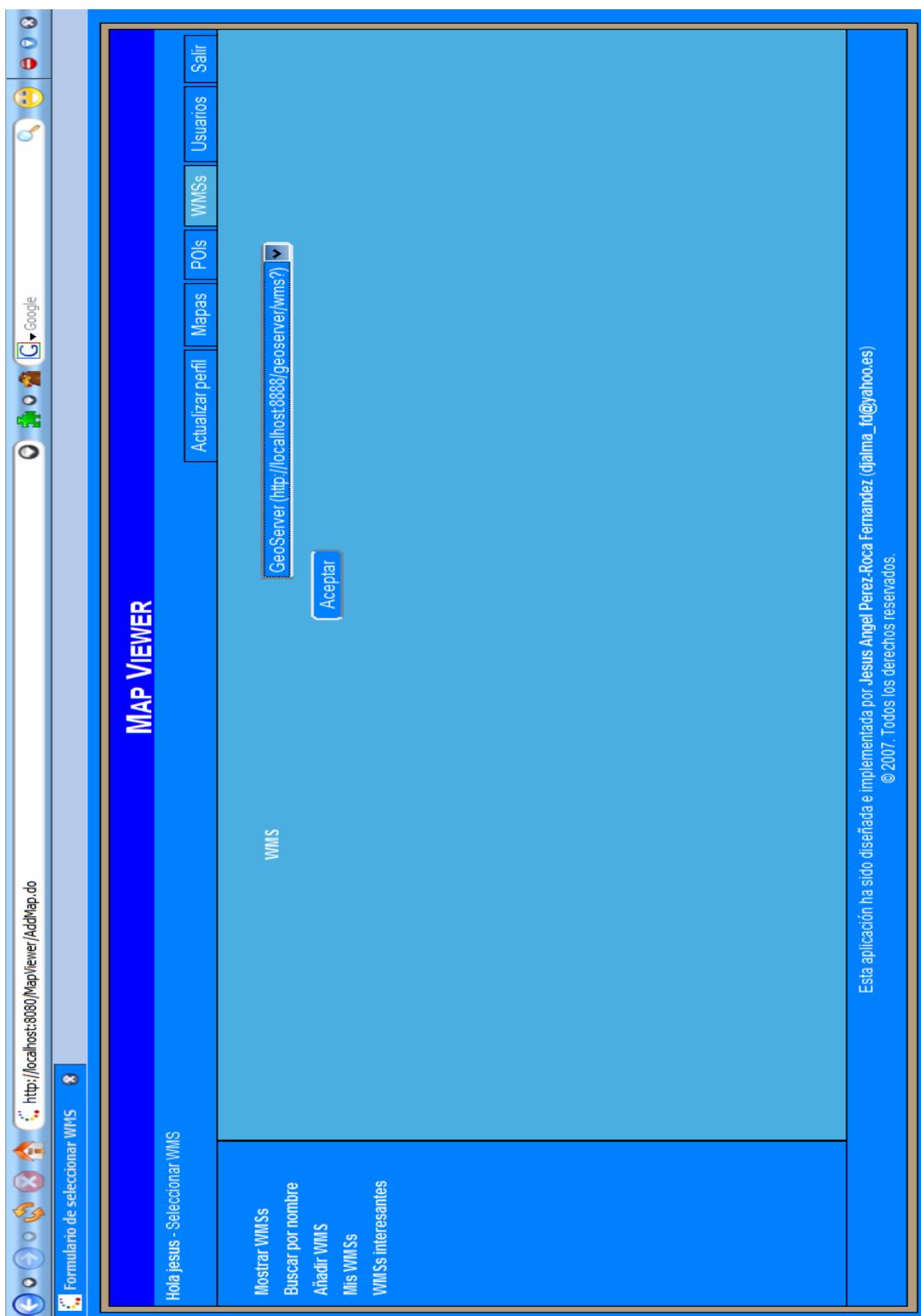


Figura 6.16: Pantalla de selección de WMS

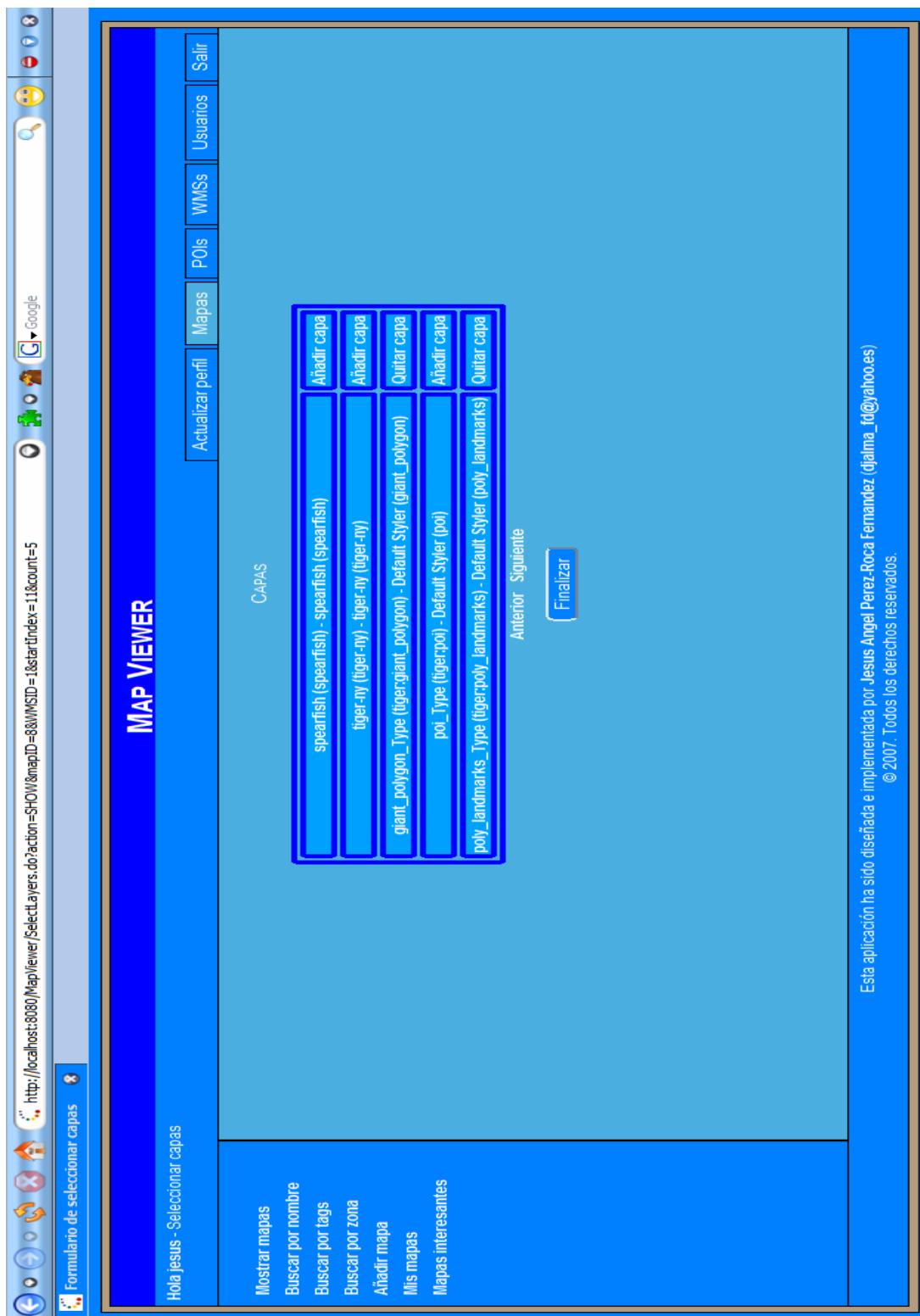


Figura 6.17: Pantalla de selección de capas

- “Buscar por nombre”, que nos permite filtrar los mapas existentes por medio de un cuadro de búsqueda, en el que introduciremos el nombre a buscar. También nos muestra los mapas en grupos de 5 con enlaces “Siguiente” y “Anterior”, en caso de ser necesarios.
- “Buscar por tags”, que nos permite filtrar los mapas existentes por medio de un cuadro de búsqueda, en el que introduciremos las etiquetas (tags) a buscar. También nos muestra los mapas en grupos de 5 con enlaces “Siguiente” y “Anterior”, en caso de ser necesarios.
- “Buscar por zona”, que nos permite filtrar los mapas existentes por medio de un cuadro de búsqueda, en el que introduciremos las coordenadas de la zona a buscar. Para facilitar esta acción podemos usar de nuevo el selector de zona (figura 6.15) que nos permite dibujar una superficie rectangular sobre un mapa del mundo. Este método también nos muestra los mapas en grupos de 5 con enlaces “Siguiente” y “Anterior”, en caso de ser necesarios.
- “Mis Mapas”, que nos muestra los mapas que hemos creado.
- “Mapas interesantes”, que nos muestra los mapas que hemos añadido a nuestra lista de favoritos.

La información obtenida con cualquiera de los métodos es análoga y nos mostrará una serie de enlaces adicionales, dependiendo de si el mapa lo hemos creado nosotros o no y de si lo tenemos en nuestra lista de mapas preferidos o no: si somos los creadores del mapa nos dejará añadir capas (nos redirige a la pantalla de selección de WMS) o borrarlo (en este caso nos devuelve a

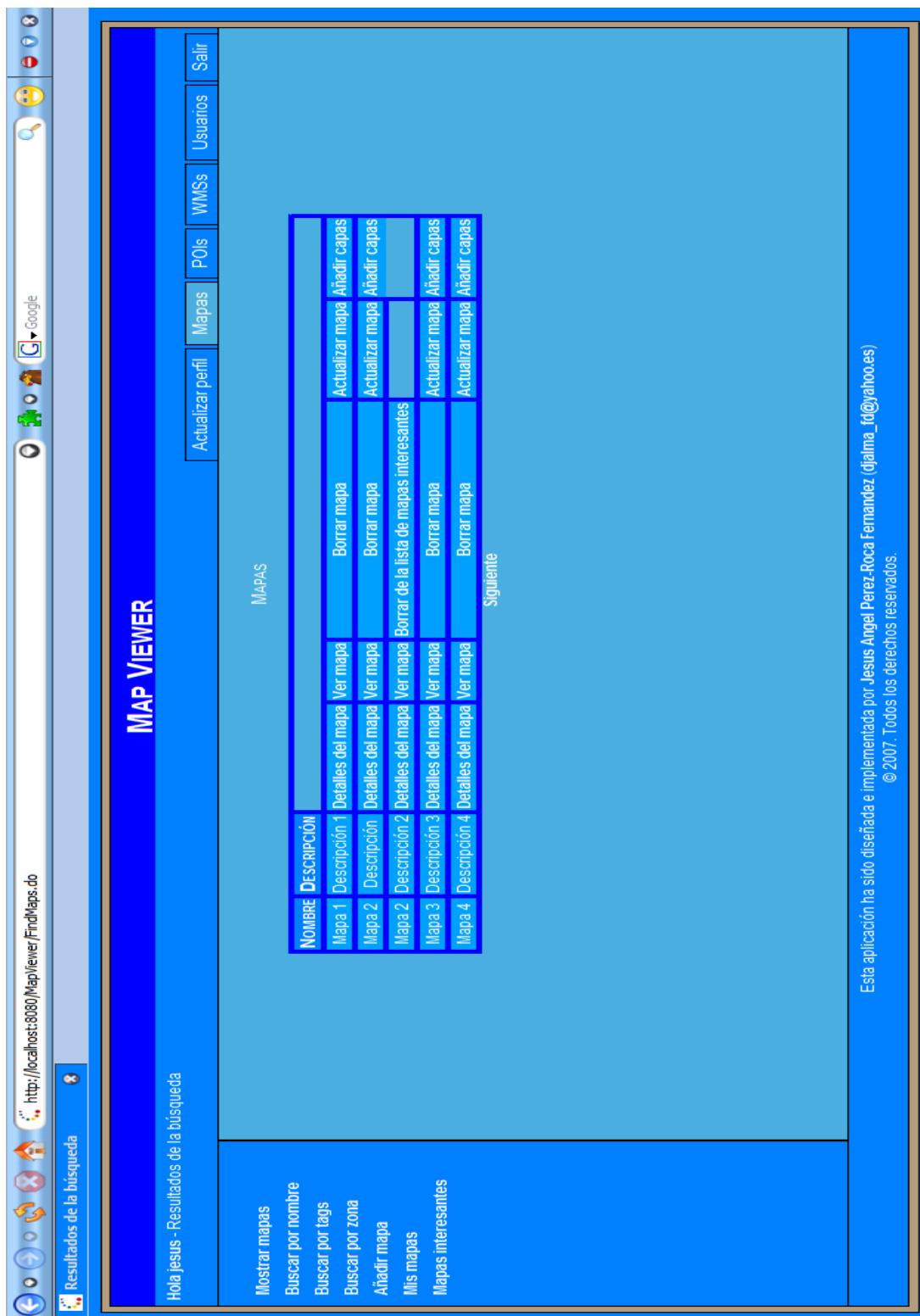


Figura 6.18: Búsqueda de mapas

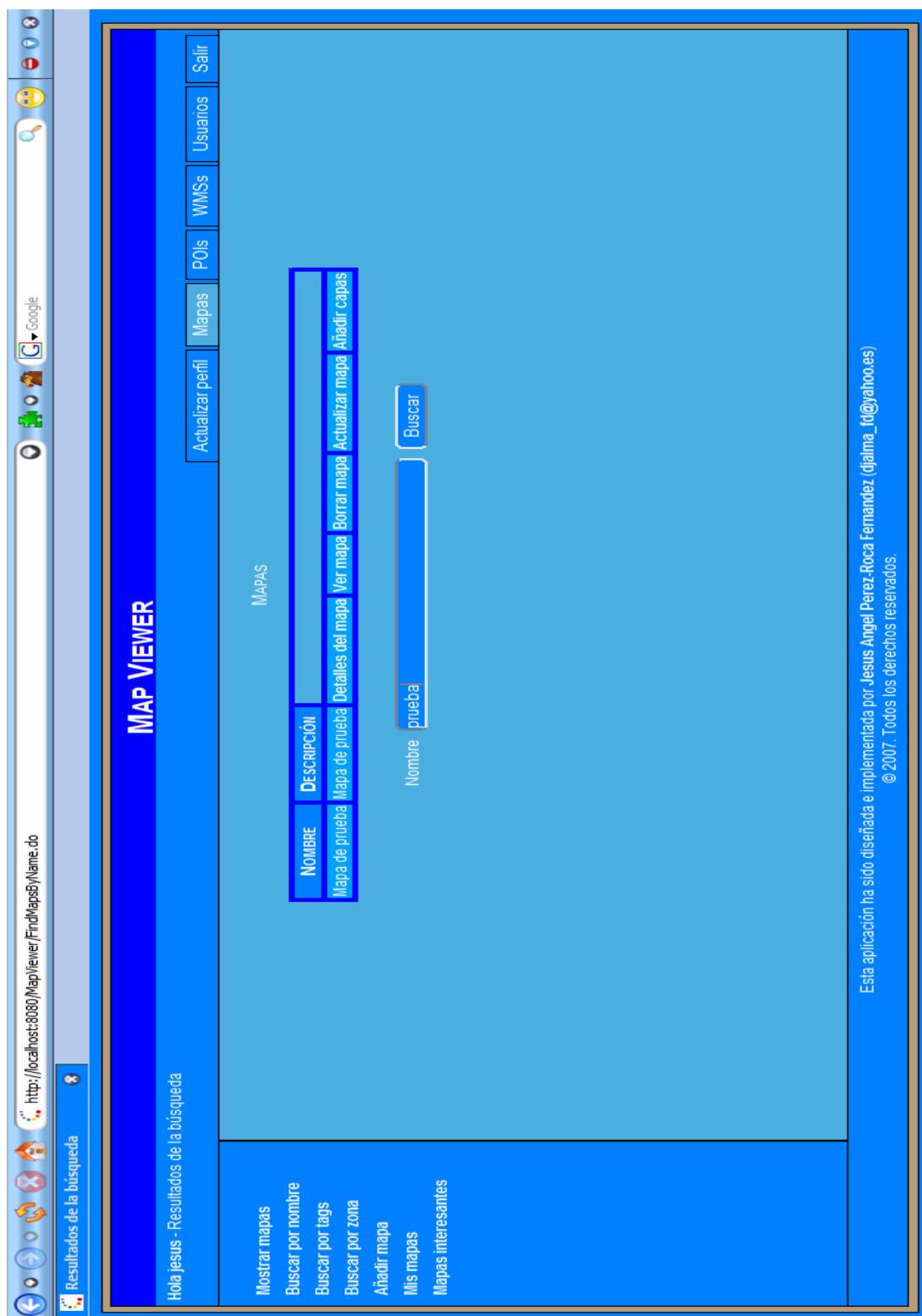


Figura 6.19: Búsqueda de mapas por nombre

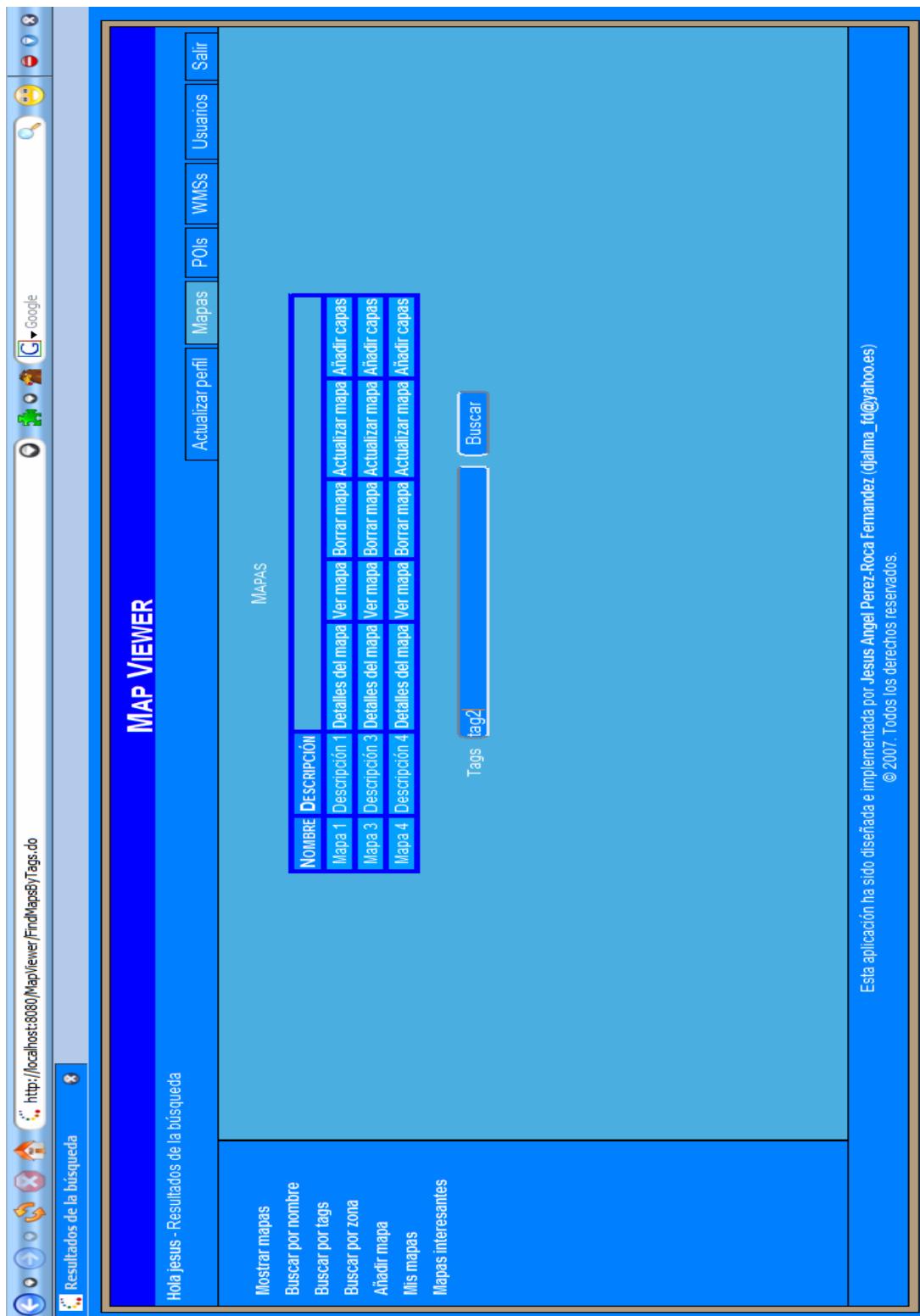


Figura 6.20: Búsqueda de mapas por etiquetas (tags)

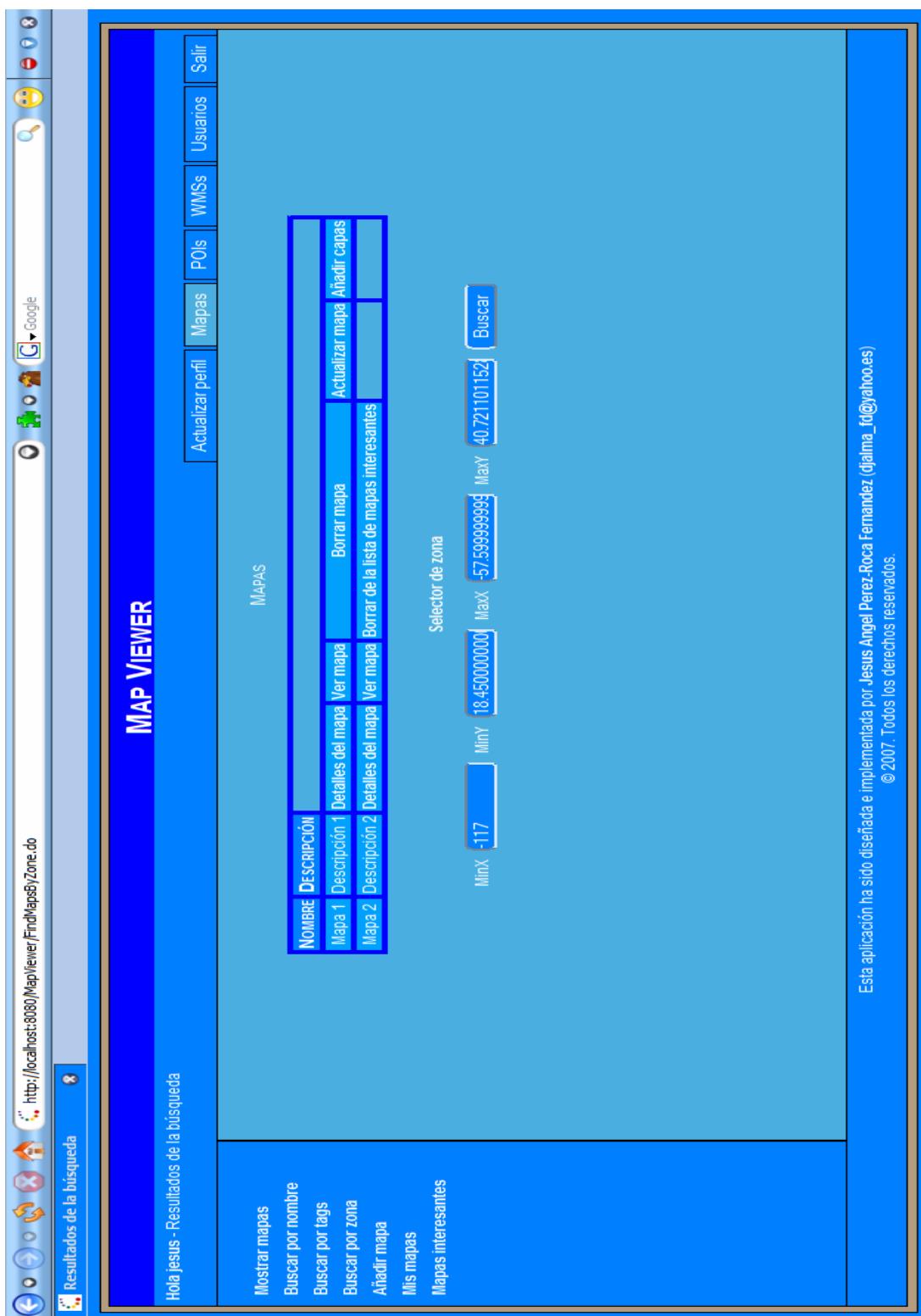


Figura 6.21: Búsqueda de mapas por zona geográfica

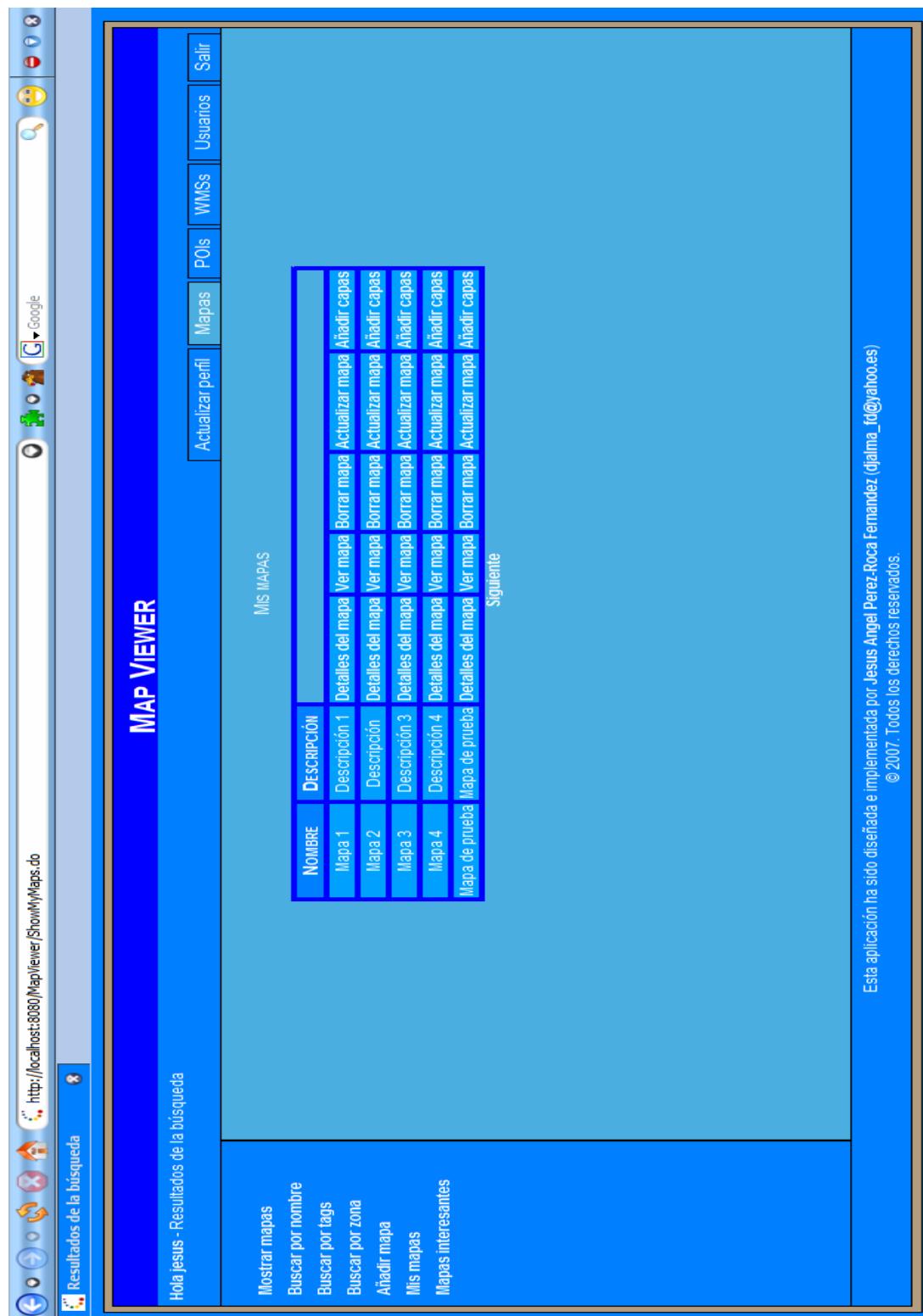


Figura 6.22: Pantalla de “Mis mapas”

la página principal) y si no lo tenemos en nuestra lista de mapas preferidos nos dejará añadirlo (y nos dejará quitarlo si ya lo tenemos en la lista), tras lo cual nos devuelve a la página principal.

En cualquier caso, siempre nos mostrará un enlace en el que podremos ver más detalladamente el mapa en cuestión. Desde esta pantalla podremos acceder a la opción de “Actualizar mapa” en caso de ser el creador del mapa. Hay que mencionar también que las etiquetas que se muestran en esta pantalla son enlaces que nos permitiría ver una lista de mapas con esa etiqueta si pulsásemos sobre ella.

Una vez localizado el mapa deseado pulsamos sobre el enlace “Ver mapa” y nos aparecerá la pantalla de visualización de mapa (figura 6.25).

Esta pantalla difiere un poco de las demás pantallas de la aplicación. La barra lateral desaparece para dejar más espacio para la visualización del mapa. Además, en la parte inferior aparecen una serie de botones que nos permiten mover por el mapa (cosa que también podemos hacer pinchando y arrastrando con el ratón), manejar el zoom (cosa que también podemos realizar con la rueda del ratón) y activar/desactivar el “Modo Insertar POI”. Si el modo está desactivado podremos movernos por el mapa libremente con el ratón y pulsar sobre los POIs (puntos de interés) que se vean en ese momento. Si pulsamos sobre un POI (punto de interés) saldrá un “popup” con la información de ese POI (punto de interés). Dicho “popup” puede moverse en caso de querer seguir viendo el mapa sin cerrarlo.

Si activamos el “Modo Insertar POI” pulsando sobre el botón correspondiente vemos que el icono del ratón cambia al entrar en la zona del mapa. Ahora

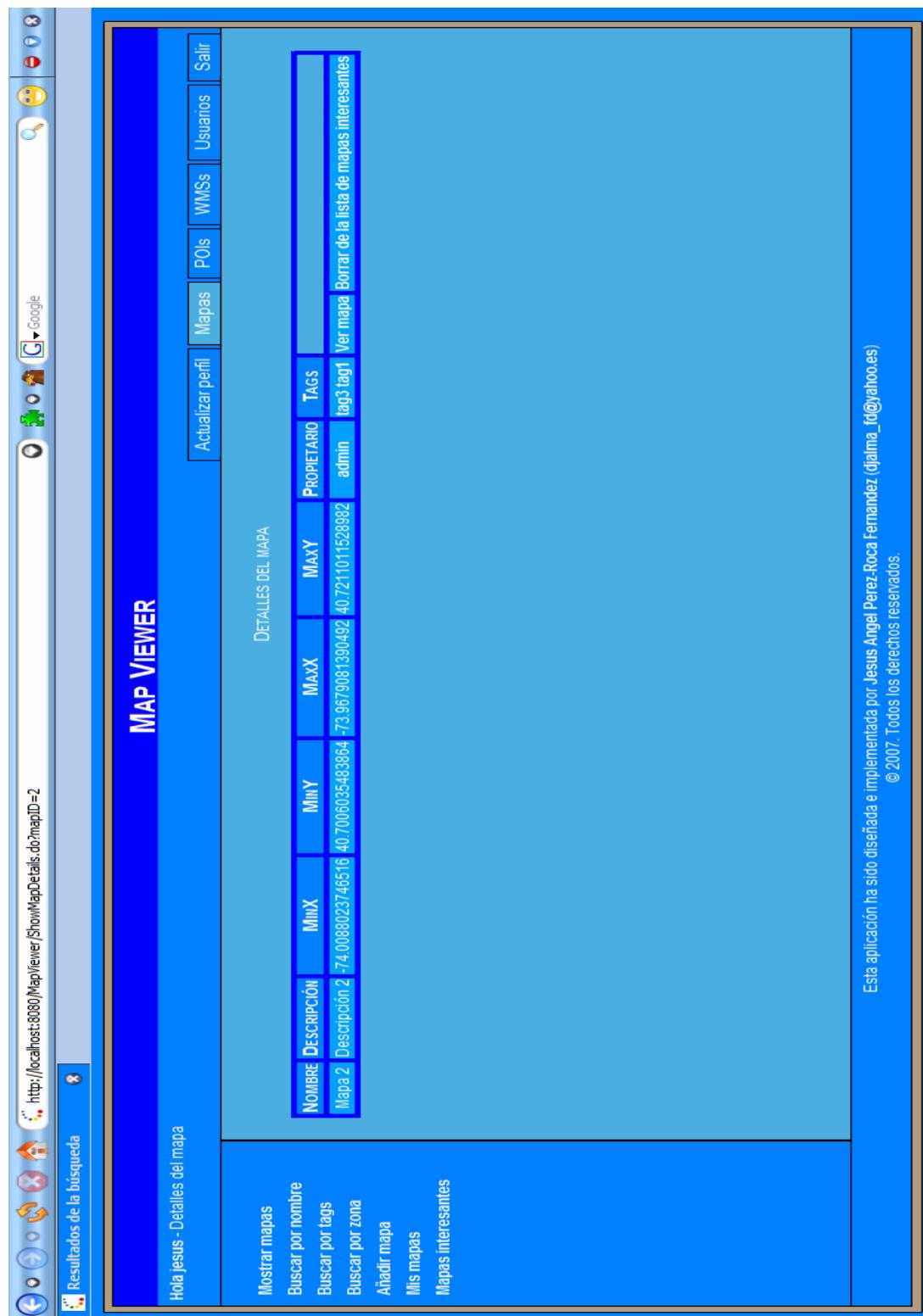


Figura 6.23: Pantalla de detalles de mapa

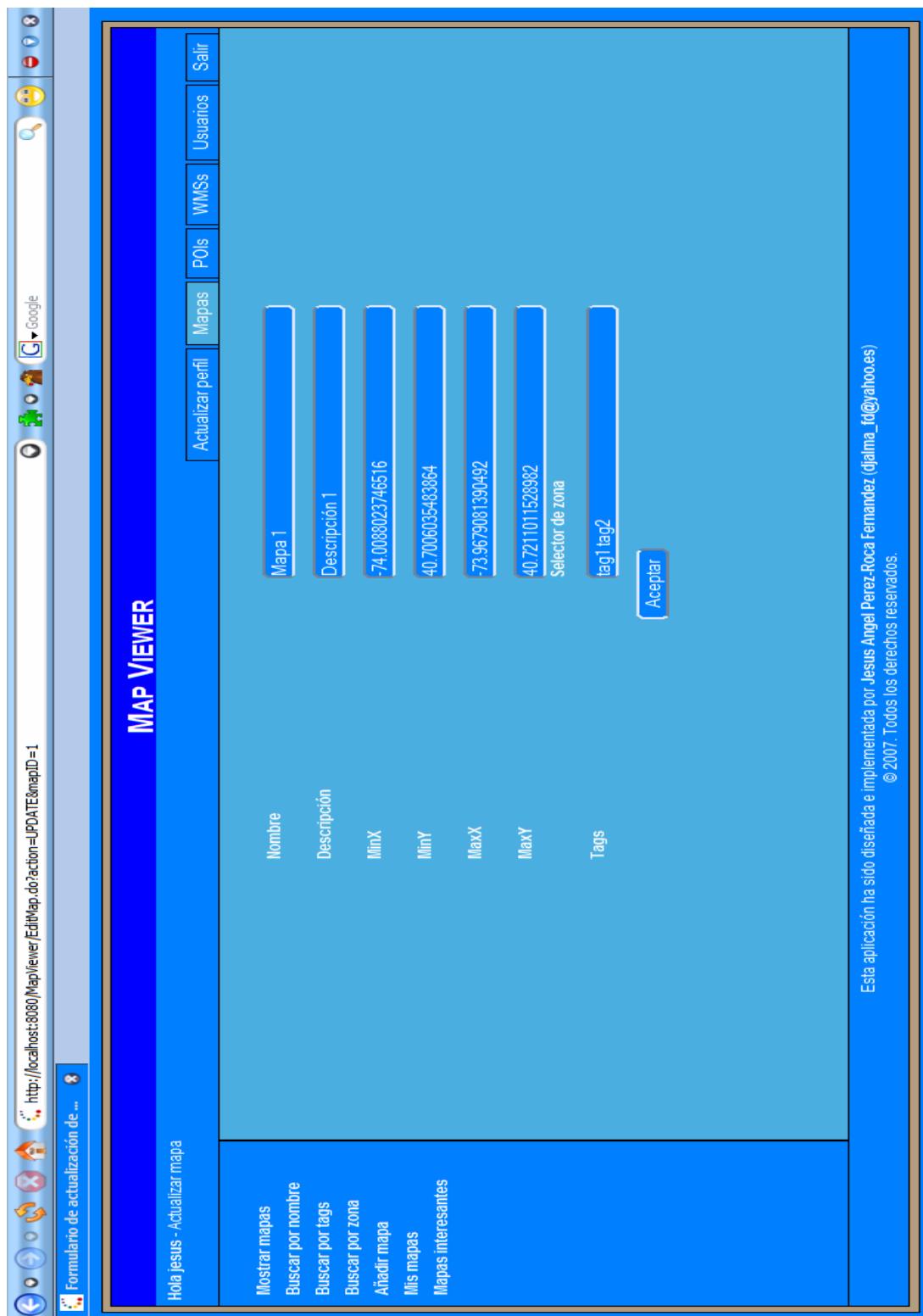


Figura 6.24: Pantalla de actualización de mapa

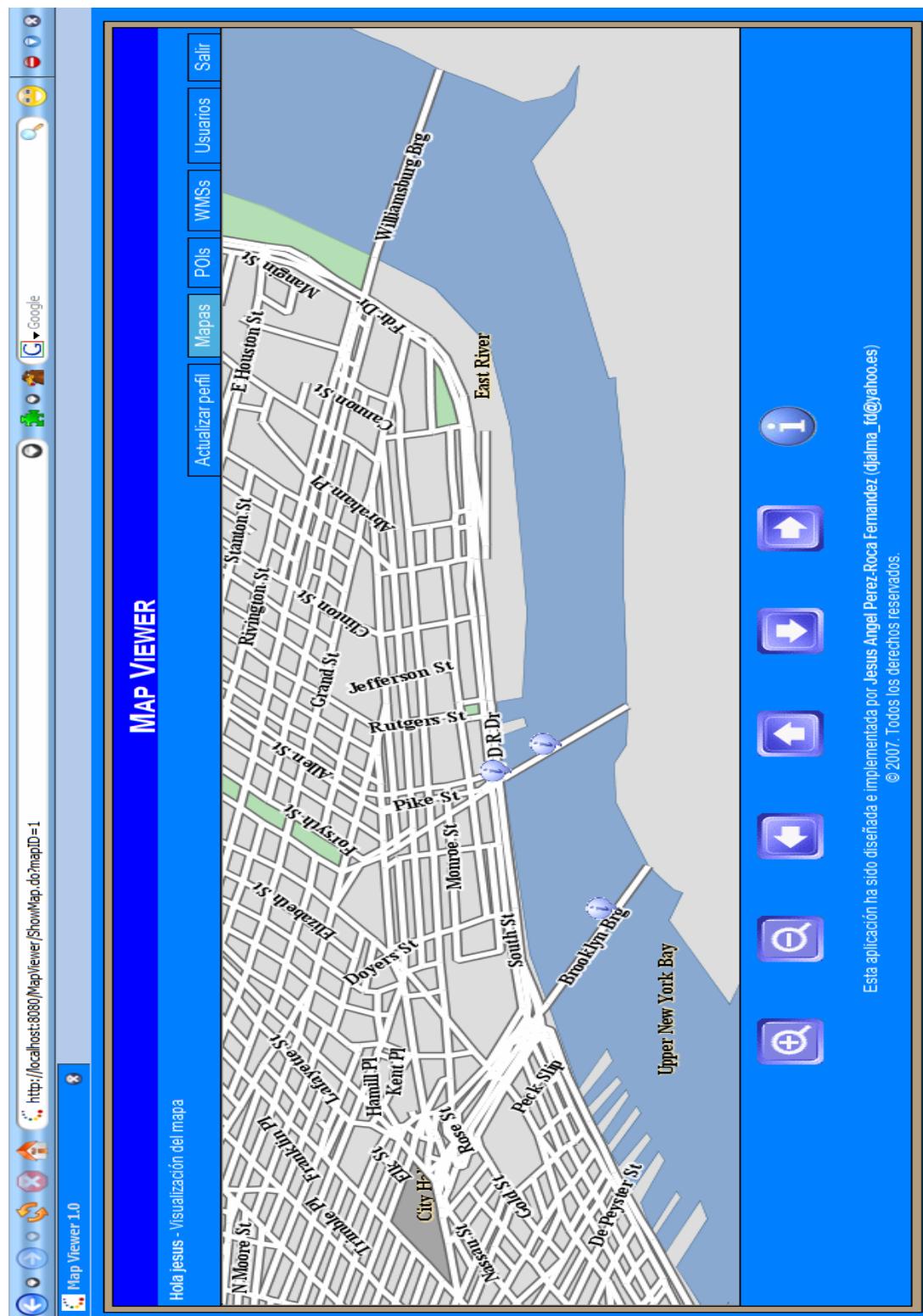


Figura 6.25: Pantalla de visualización de mapa

si pulsamos sobre el mapa podremos añadir en esas coordenadas un POI (punto de interés). Antes de llevarnos a la pantalla de “Añadir POI” nos sale un “popup” que nos permite cancelar la acción en caso de haber pulsado en unas coordenadas incorrectas.

Si pulsamos en el botón “Aceptar” nos aparece la pantalla de “Añadir POI” con los campos de coordenadas ya cubiertos, por lo que sólo deberemos ponerle un nombre, una descripción, una serie de etiquetas (tags) para facilitar la posterior búsqueda del POI (punto de interés) y el código HTML que queremos mostrar en el “popup” que saldrá al pulsar sobre el POI (punto de interés) al visualizarlo en un mapa. Para facilitar la creación del código HTML, se ha incluido un sencillo editor HTML, que permite insertar fácilmente imágenes o enlaces, así como varias opciones de edición de texto y otras opciones no tan habituales en una descripción normal de un punto de interés. Una vez introducidos todos los datos pulsaremos el botón “Aceptar” y volveremos a la página principal, pero con las opciones relativas a los POIs (puntos de interés).

Desde ahí podremos comprobar que nuestro POI (punto de interés) se creó correctamente. Para ello existen varios métodos de búsqueda:

- “Mostrar POIs”, que nos muestra todos los POIs existentes en la base de datos (en grupos de 5 con enlaces “Siguiente” y “Anterior”, en caso de ser necesarios).
- “Buscar por nombre”, que nos permite filtrar los POIs existentes por medio de un cuadro de búsqueda, en el que introduciremos el nombre a buscar. También nos muestra los POIs en grupos de 5 con enlaces

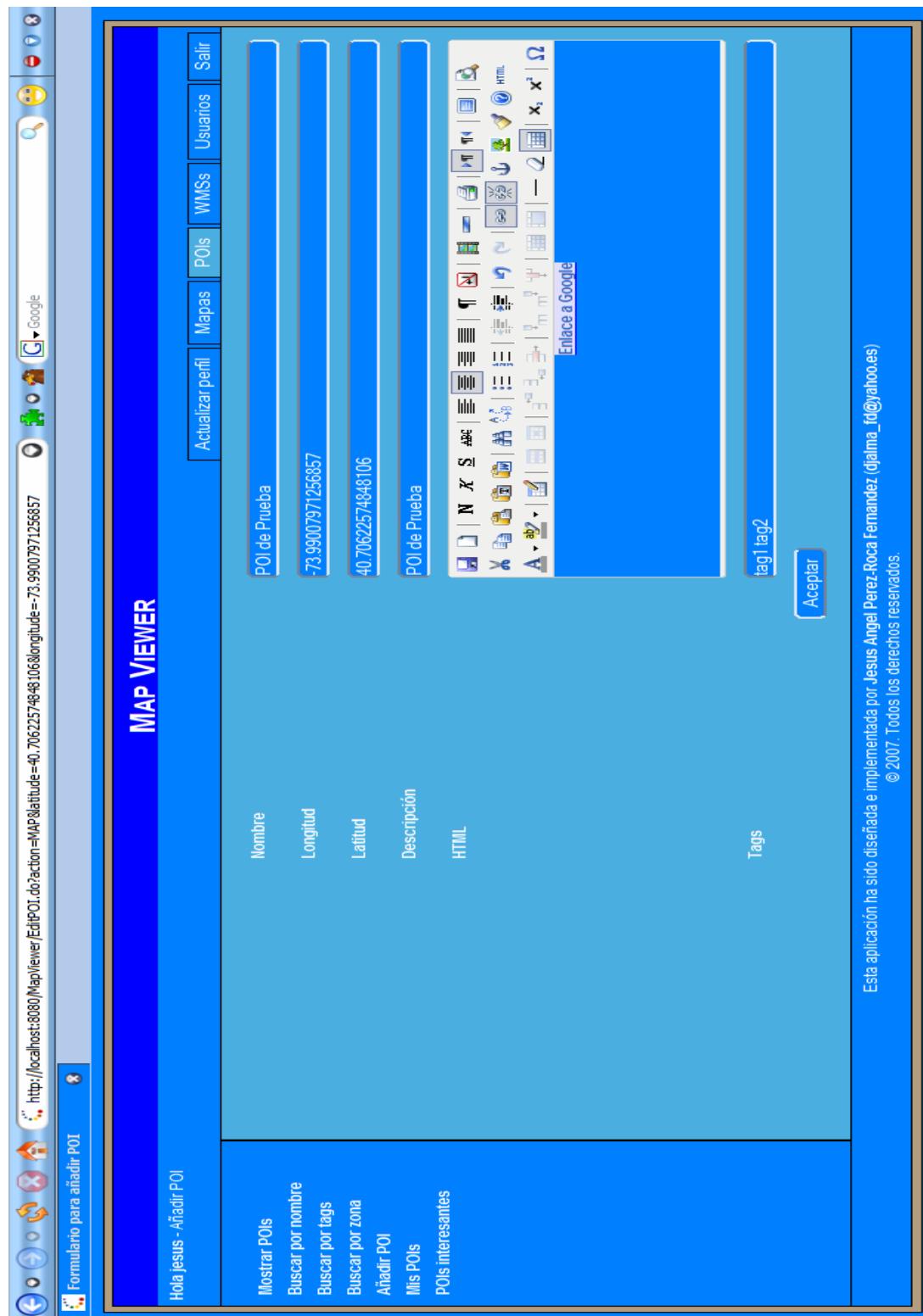


Figura 6.26: Pantalla de añadir POI (punto de interés)

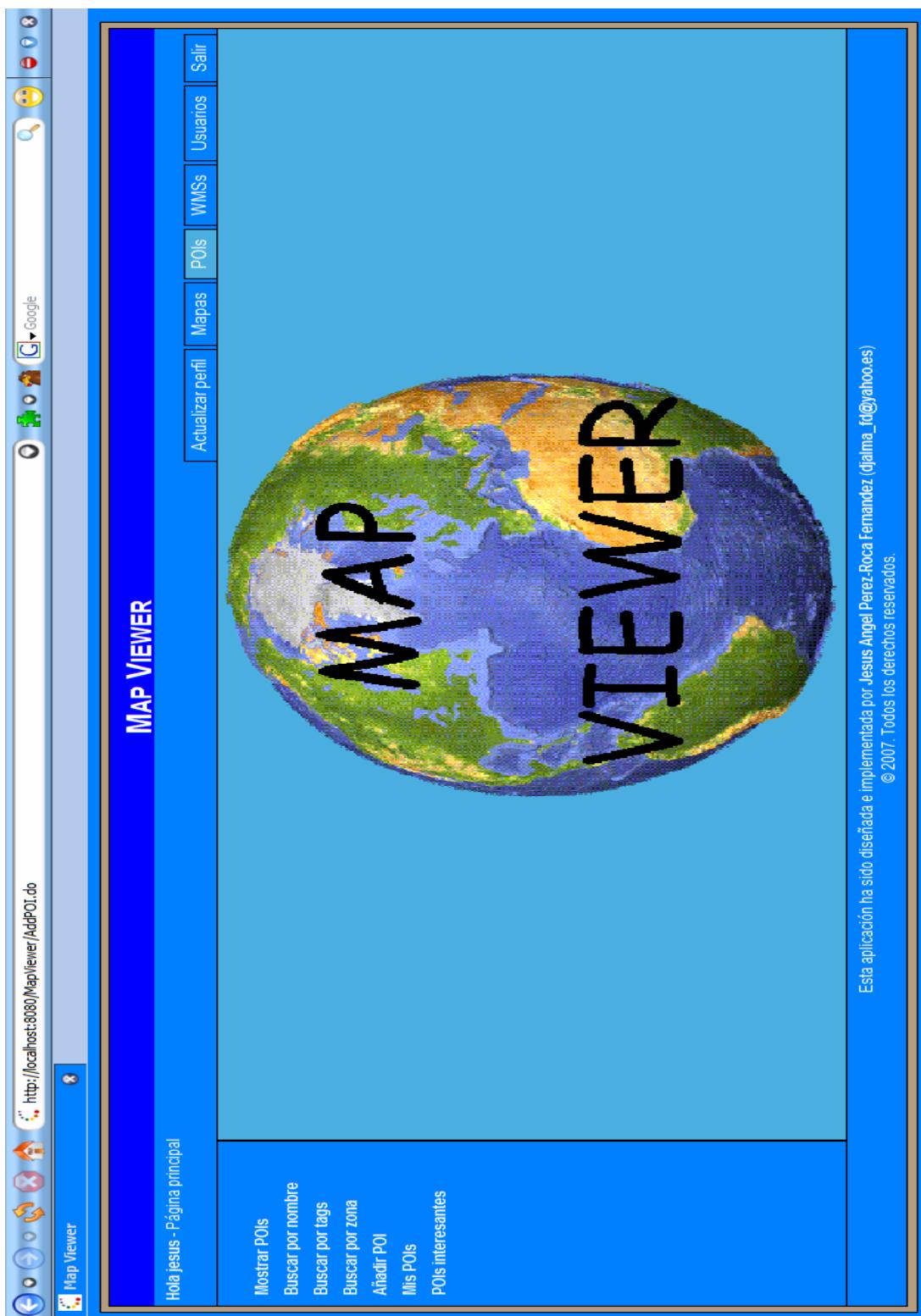


Figura 6.27: Pantalla principal (opciones POIs)

“Siguiente” y “Anterior”, en caso de ser necesarios.

- “Buscar por tags”, que nos permite filtrar los POIs existentes por medio de un cuadro de búsqueda, en el que introduciremos las etiquetas (tags) a buscar. También nos muestra los POIs en grupos de 5 con enlaces “Siguiente” y “Anterior”, en caso de ser necesarios.
- “Buscar por zona”, que nos permite filtrar los POIs existentes por medio de un cuadro de búsqueda, en el que introduciremos las coordenadas de la zona a buscar. Para facilitar esta acción podemos usar de nuevo el selector de zona que nos permite dibujar una superficie rectangular sobre un mapa del mundo. Este método también nos muestra los POIs en grupos de 5 con enlaces “Siguiente” y “Anterior”, en caso de ser necesarios.
- “Mis POIs”, que nos muestra los POIs que hemos creado.
- “POIs interesantes”, que nos muestra los POIs que hemos añadido a nuestra lista de favoritos.

La información obtenida con cualquiera de los métodos es análoga y nos mostrará una serie de enlaces adicionales, dependiendo de si el POI lo hemos creado nosotros o no y de si lo tenemos en nuestra lista de POIs preferidos o no: si somos los creadores del POI nos dejará actualizarlo o borrarlo (en este caso nos devuelve a la página principal) y si no lo tenemos en nuestra lista de POIs preferidos nos dejará añadirlo (y nos dejará quitarlo si ya lo tenemos en la lista), tras lo cual nos devuelve a la página principal.

En cualquier caso, siempre nos mostrará un enlace en el que podremos ver más detalladamente el POI en cuestión. Las etiquetas que se muestran en

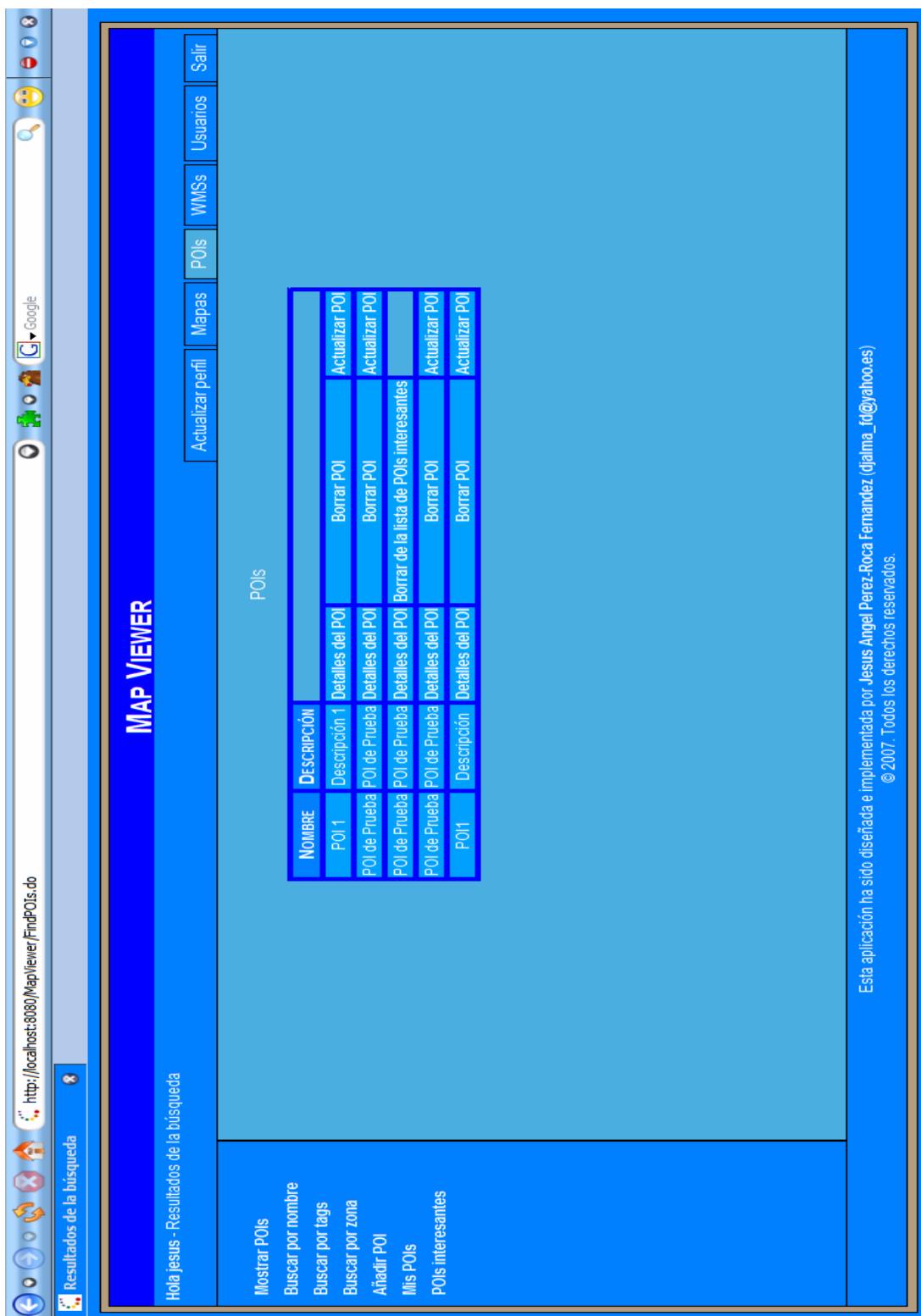


Figura 6.28: Búsqueda de POIs (puntos de interés)

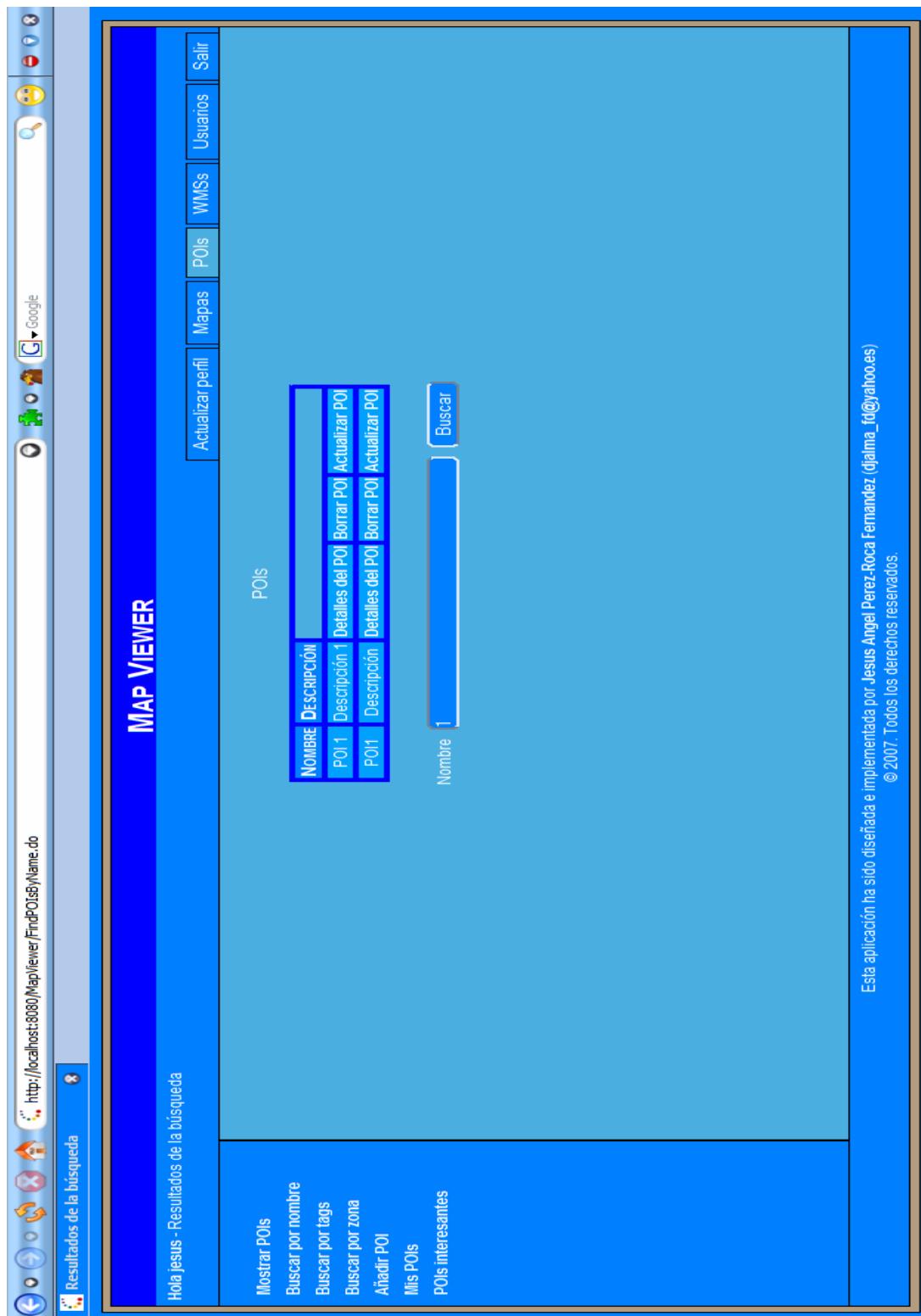


Figura 6.29: Búsqueda de POIs (puntos de interés) por nombre

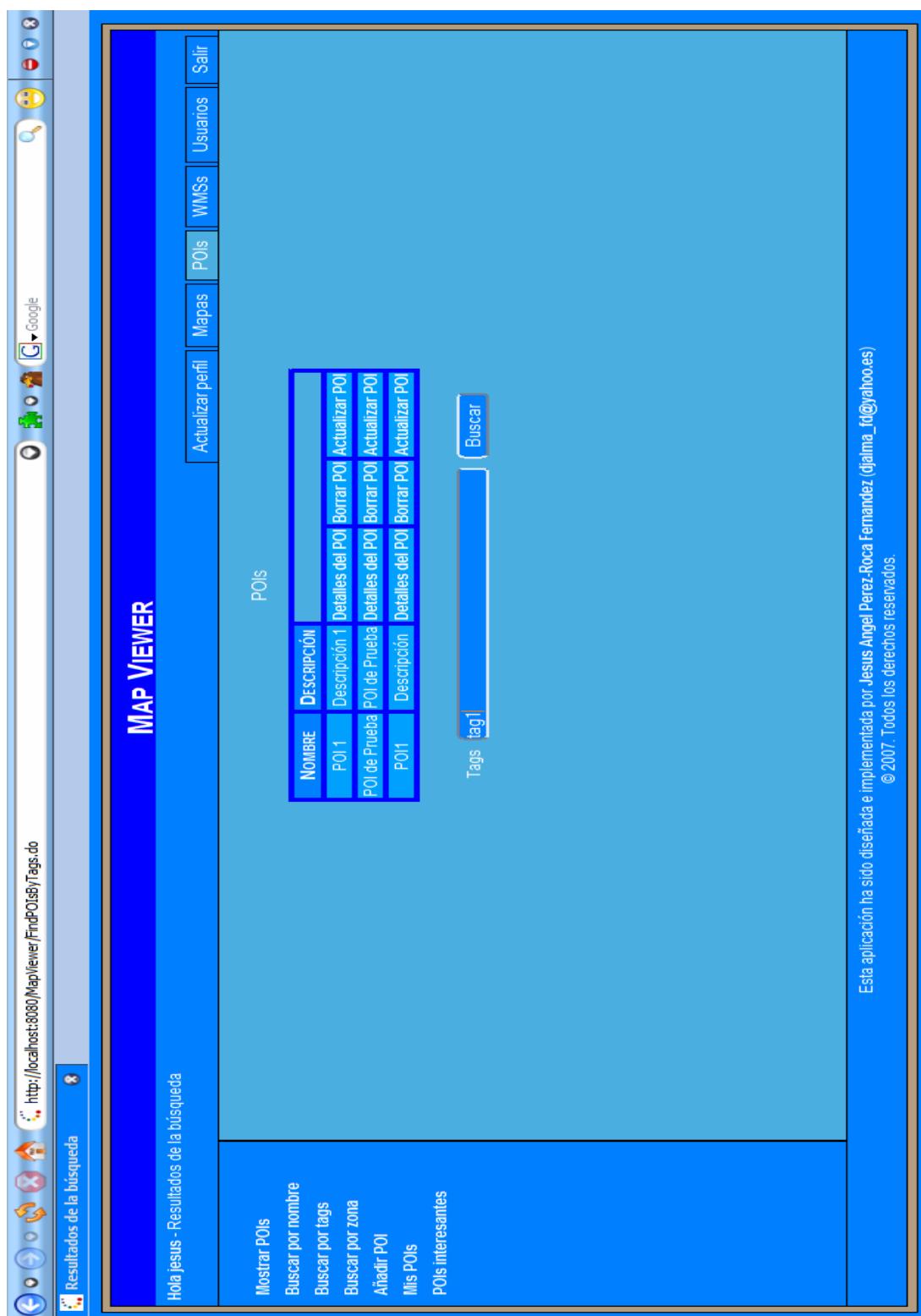


Figura 6.30: Búsqueda de POIs (puntos de interés) por etiquetas (tags)

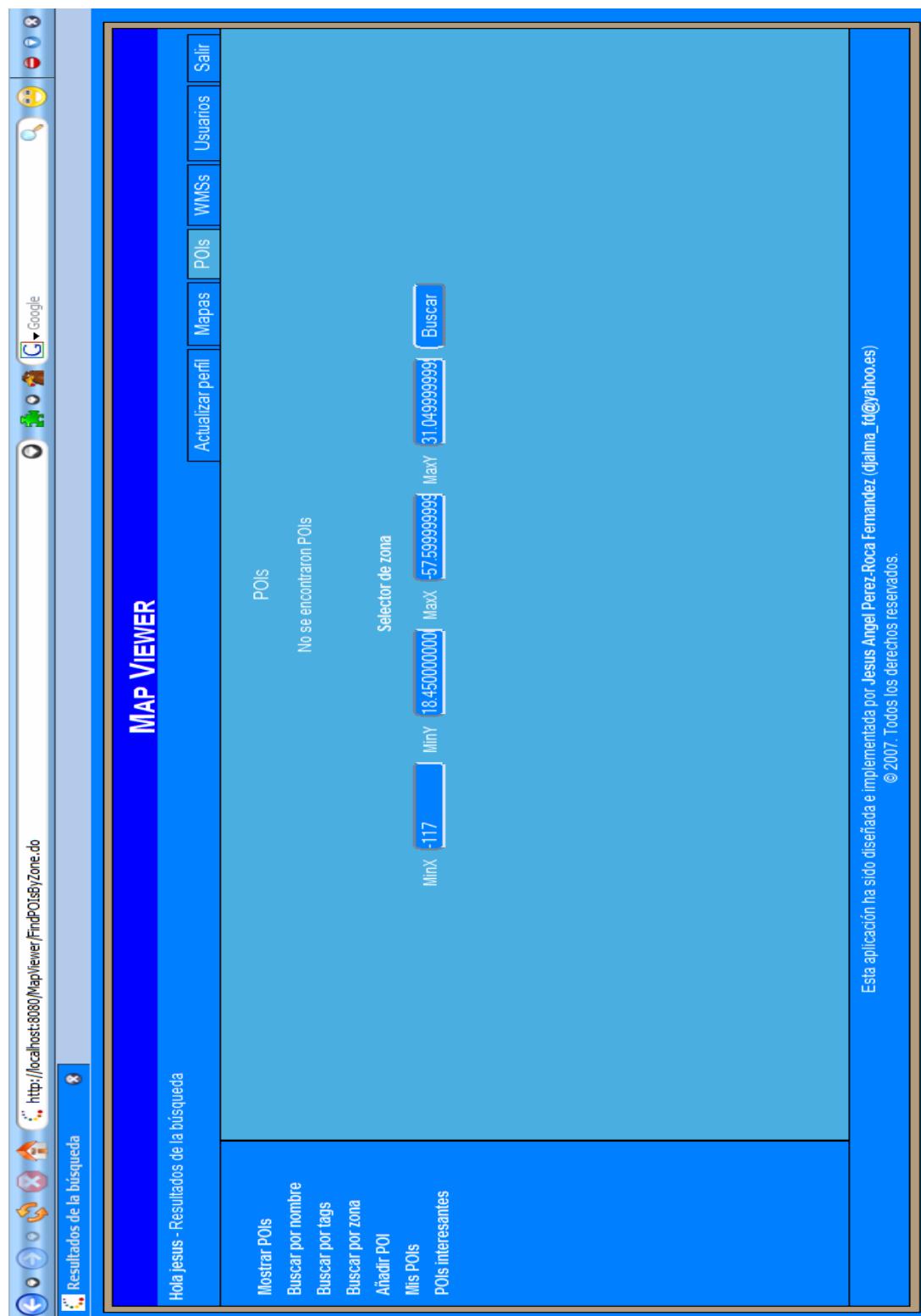


Figura 6.31: Búsqueda de POIs (puntos de interés) por zona geográfica

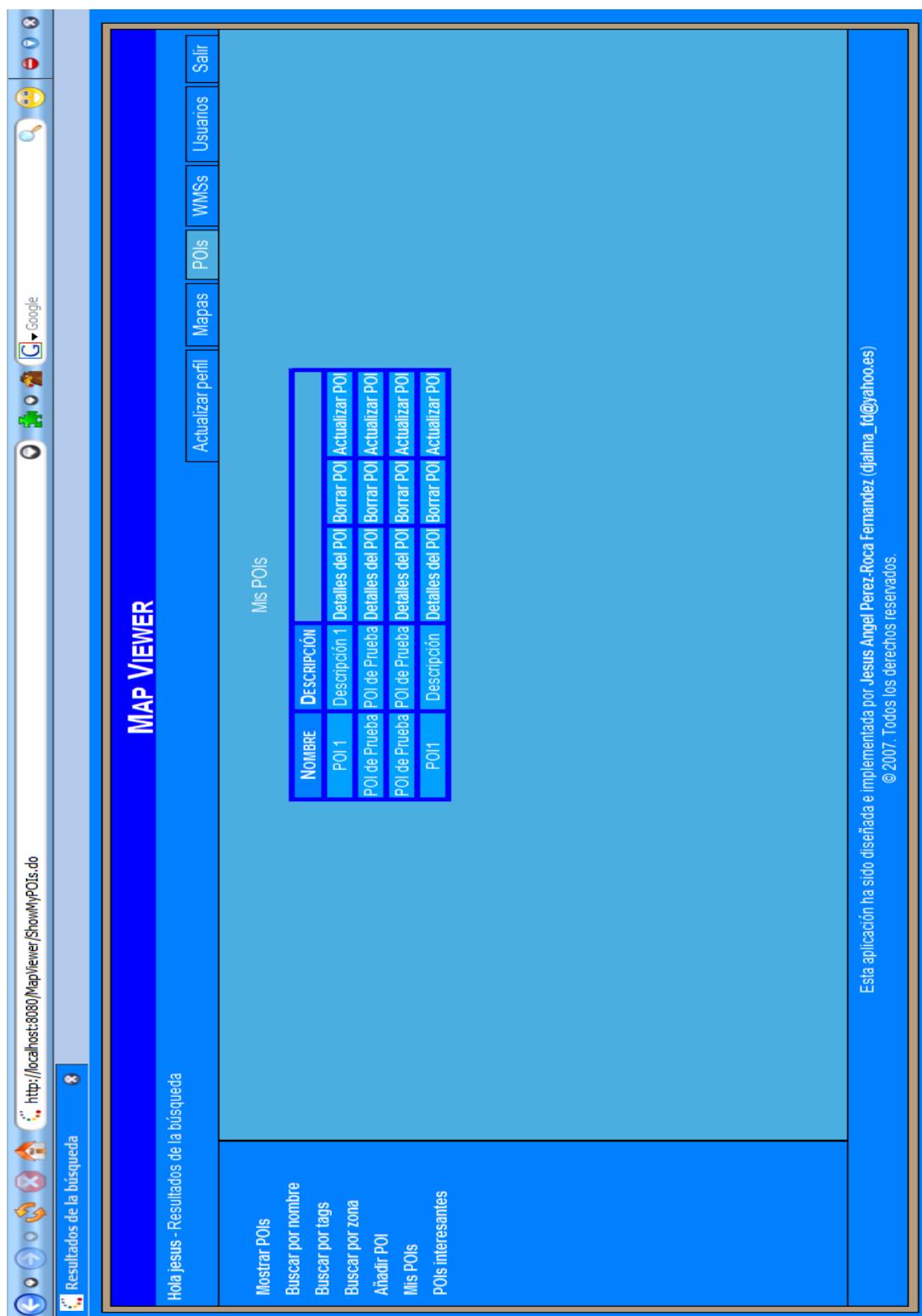


Figura 6.32: Pantalla de “Mis POIs”

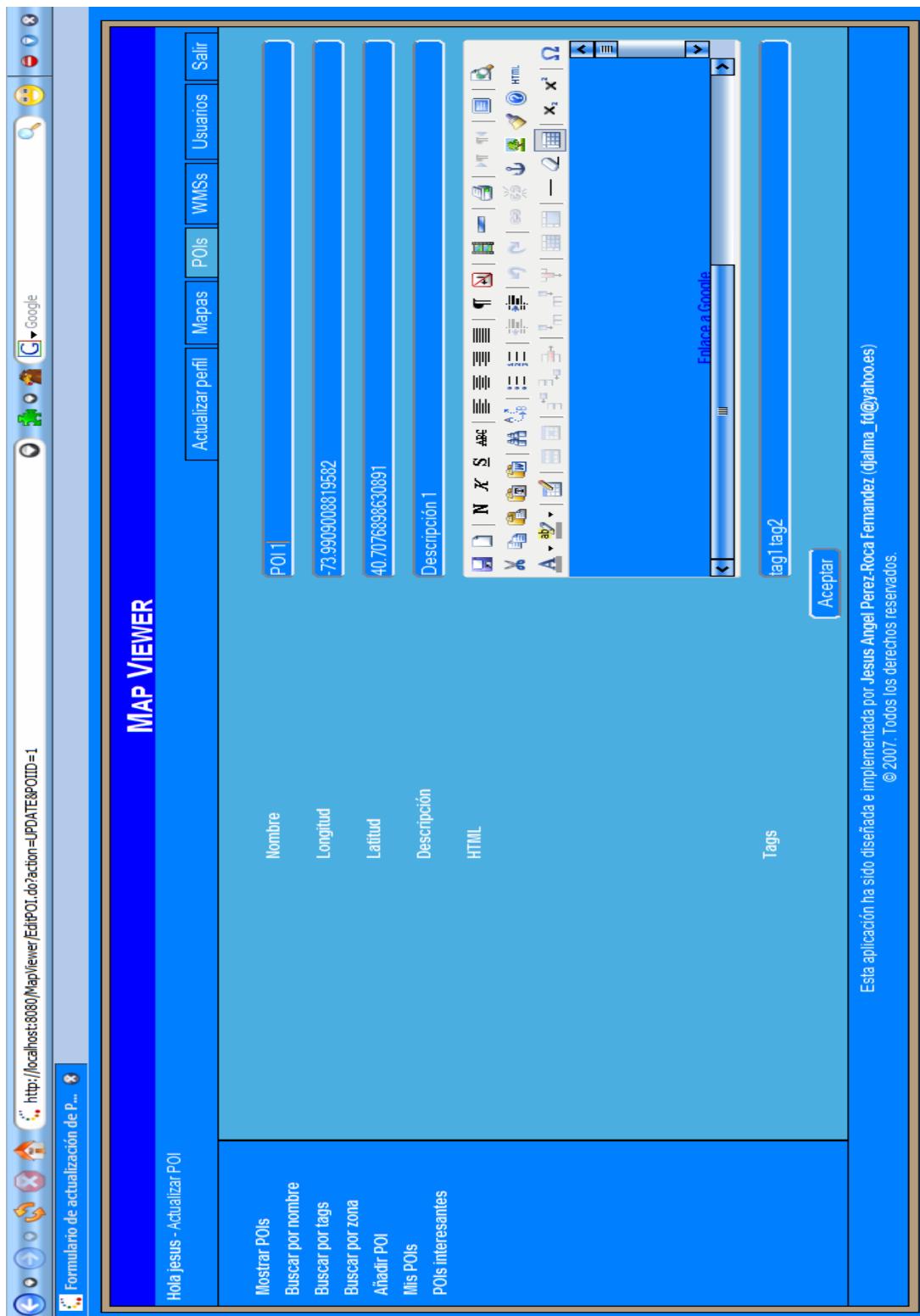


Figura 6.33: Pantalla de actualización de POI (punto de interés)

esta pantalla son enlaces que nos permitiría ver una lista de POIs con esa etiqueta si pulsásemos sobre ella.

Por último, podemos ver una lista de los usuarios registrados en la aplicación para poder acceder fácilmente a los mapas, WMSs y POIs que crearon. Para ello primero deberemos pulsar sobre el botón “Usuarios” de la barra de botones, para que nos muestre las opciones relativas a los usuarios.

Existen dos métodos para buscar usuarios:

- “Mostrar usuarios”, que nos muestra todos los usuarios existentes en la base de datos (en grupos de 5 con enlaces “Siguiente” y “Anterior”, en caso de ser necesarios).
- “Buscar por identificador de usuario”, que nos permite filtrar los usuarios existentes por medio de un cuadro de búsqueda, en el que introduciremos el identificador de usuario (login) a buscar. También nos muestra los usuarios en grupos de 5 con enlaces “Siguiente” y “Anterior”, en caso de ser necesarios.

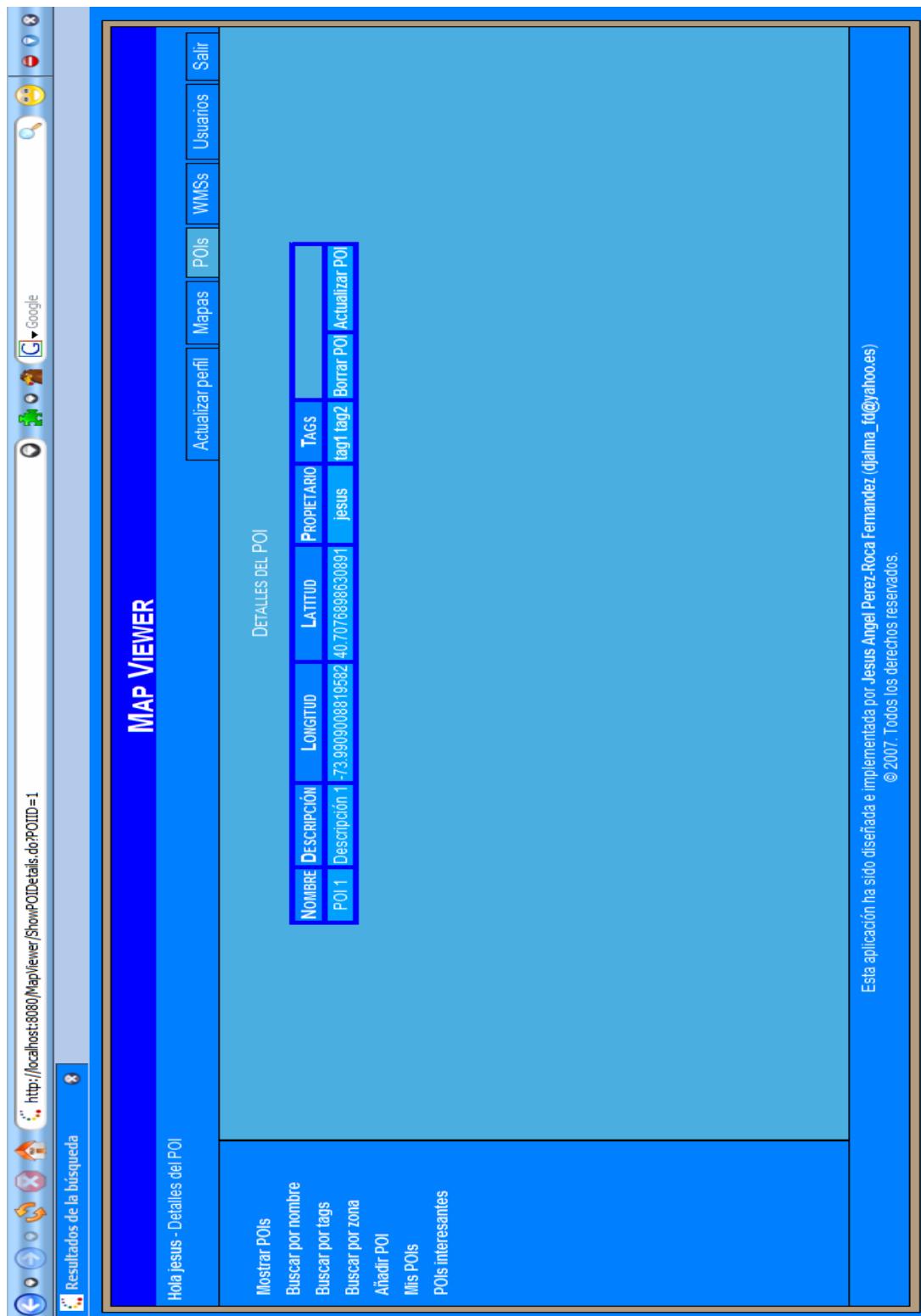


Figura 6.34: Pantalla de detalles de POI (punto de interés)

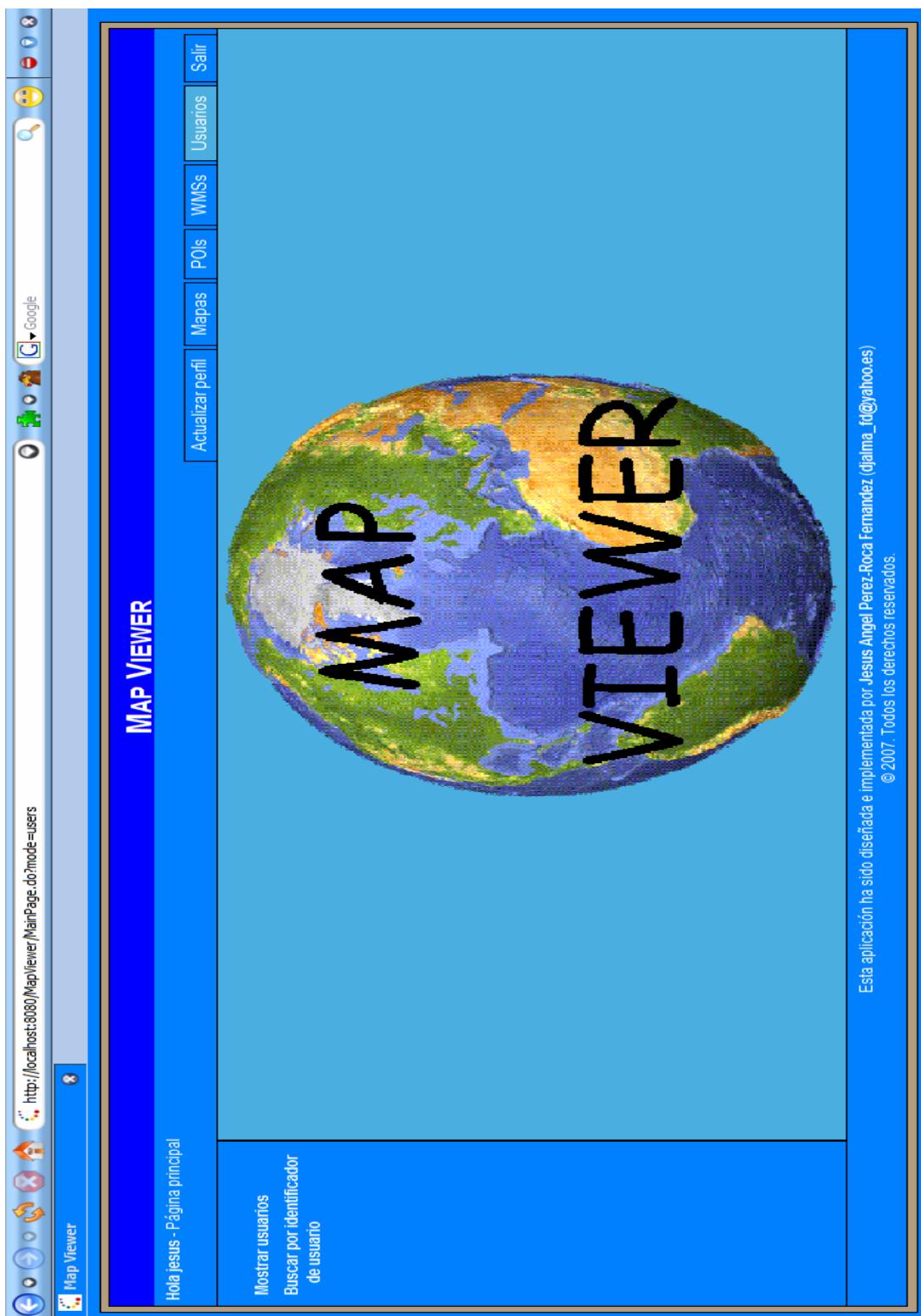


Figura 6.35: Pantalla principal (opciones usuarios)

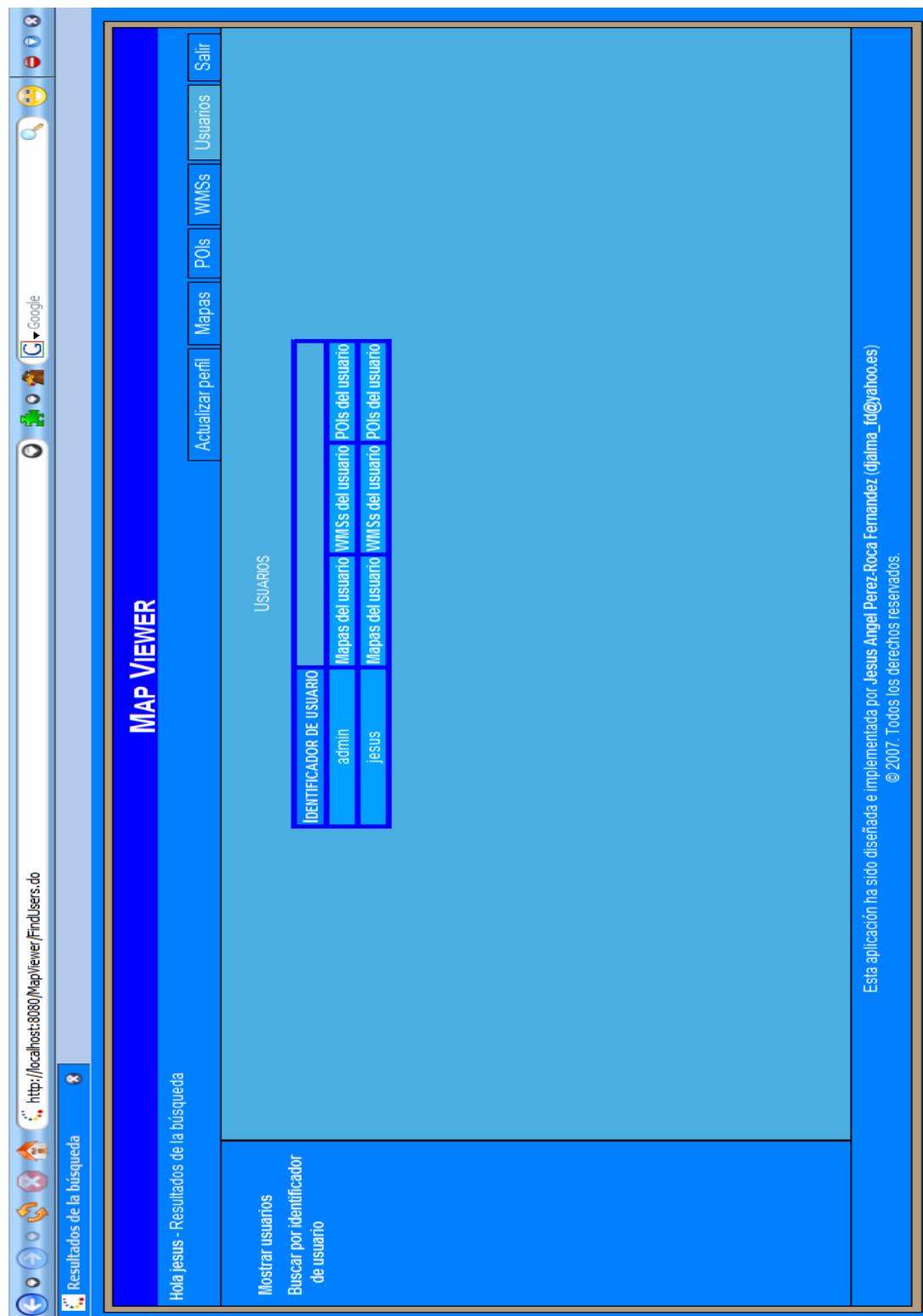


Figura 6.36: Búsqueda de usuarios

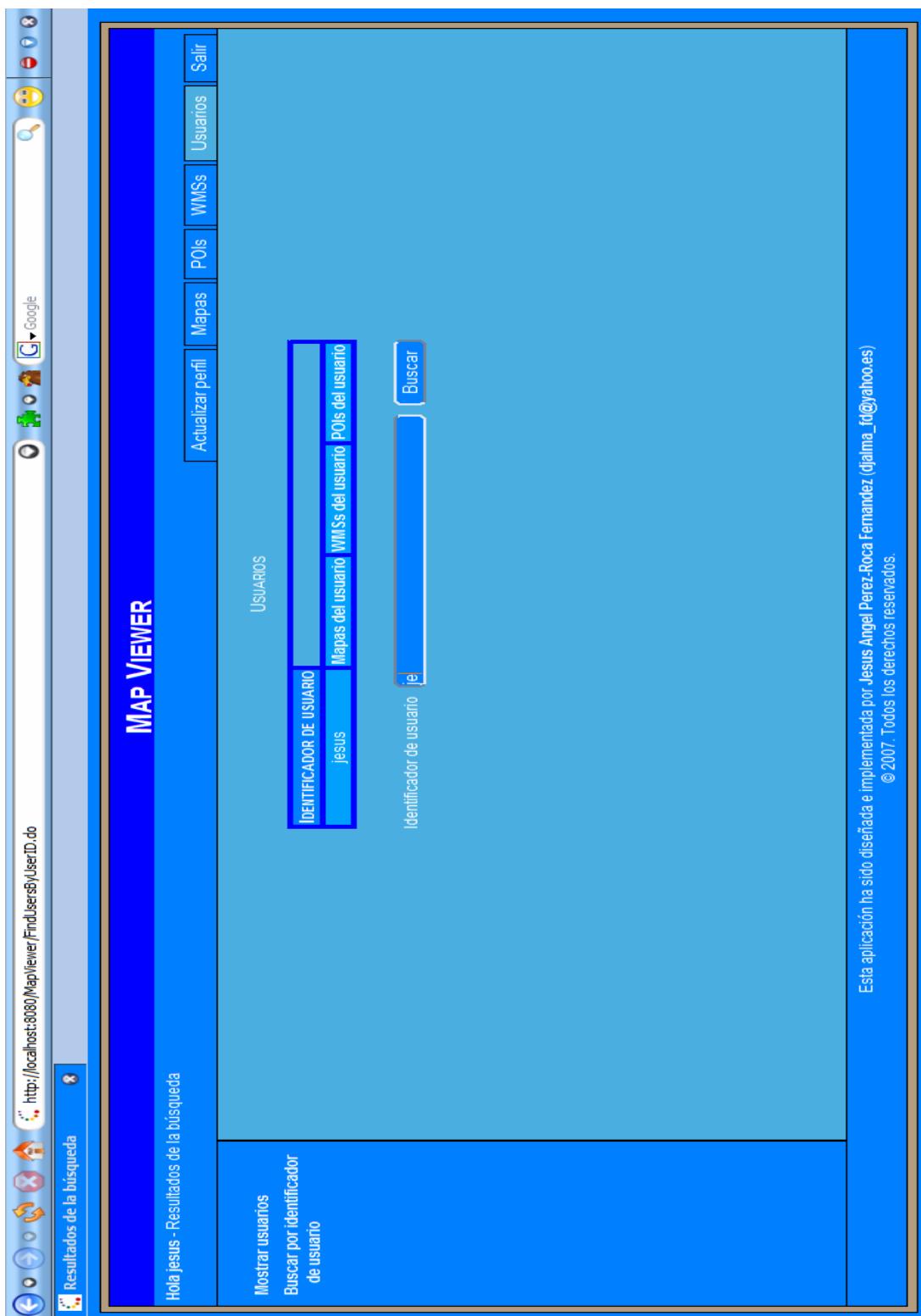


Figura 6.37: Búsqueda de usuarios por login (identificador de usuario)



## Conclusiones y trabajo futuro

A continuación se muestran las principales líneas de mejora que podrían llevarse a cabo en un futuro.

- **Creación de una aplicación de administración:** Se podría crear una sencilla aplicación de administración de la aplicación pudea borrar usuarios o administrar más fácilmente la aplicación. En estos momentos, el usuario “admin” puede borrar mapas, WMSs o puntos de información de otros usuarios, pero tiene que hacerlo uno a uno ya que no existe una opción de borrado en masa, que podría ser deseable en un futuro.
- **Internacionalización de la aplicación:** La aplicación está pensada para que sea fácil su internacionalización, ya que para mostrar los mensajes se hace uso del soporte para la internacionalización de Struts, que utiliza un fichero .properties con el contenido de los mensajes en un determinado idioma. Si se quisiera añadir, por ejemplo, el idioma inglés lo único que debería hacerse es traducir el contenido del fichero *Messages.properties* y grabarlo como *Messages\_en.properties*, así como implementar una manera de escoger el idioma de la aplicación.

- **Mejora del rendimiento:** Se podría intentar mejorar el rendimiento a la hora de mostrar los mapas, haciendo que, en lugar de dividir la zona a mostrar en nueve subzonas y hacer nueve peticiones en paralelo, podría dividirse en más zonas. También se podría hacer que el renderizado del mapa con las imágenes resultantes sea más efectivo.
- **Internacionalización de la base de datos:** Se podría hacer que el contenido de la base de datos esté disponible en varios idiomas, para que todo el contenido visual de la aplicación web sea internacionalizable en los idiomas soportados.
- **Añadir nuevas funcionalidades:** Por último, se podrían añadir nuevas funcionalidades, tales como el uso de RSS para obtener los últimos mapas, POIs o WMSs incluidos, o mejorar la interfaz de la aplicación.



## Manual de instalación

A continuación se enumerará el software necesario para la ejecución de la aplicación y se proporcionará una breve explicación de cómo instalar el mismo. El desarrollo del software se ha llevado a cabo en el sistema operativo Microsoft Windows XP, aunque las instrucciones necesarias para realizar la instalación en sistemas operativos basados en Unix no difieren demasiado y bastaría con utilizar las versiones Unix de los programas utilizados.

Las instrucciones para Microsoft Windows XP funcionan también en las ediciones 95, 98 y ME, pero hay que tener en cuenta que ciertos programas pueden no funcionar correctamente si se instalan bajo directorios que contengan espacios en el nombre o nombres demasiado largos.

Además se recomienda que tras instalar un programa, se creen las variables de entorno necesarias para acceder al directorio de instalación del mismo y para incluir el programa en la variable de entorno PATH. Esto ayudará al usuario a realizar las operaciones de forma más sencilla. Las variables de entorno se han de crear de forma persistente, es decir, en el archivo */.bash\_profile* para los sistemas operativos basados en UNIX, en el archivo *autoexec.bat* en Microsoft Windows 95/98/ME o en el menú de variables de entorno que

se puede encontrar en el menú *Sistema* del Panel De Control de Microsoft Windows XP.

### **A.1. Instalación de Java Runtime Environment**

Es necesario disponer de una máquina virtual Java para el correcto funcionamiento de la aplicación, así como de parte del software necesario para la compilación e instalación de la misma. En el CD adjunto se incluye el Java JDK 1.6.0, aunque llegaría con tener instalada la versión 1.5.0 o superior de la misma para poder compilar y ejecutar el proyecto.

### **A.2. Instalación de PostgreSQL**

En el CD adjunto se incluye un instalador de fácil manejo, así como el driver JDBC de PostgreSQL. Además, si no se quieren modificar los archivos del código fuente más de lo estrictamente necesario, habría que crear un usuario llamado “postgres” con contraseña “postgres”.

### **A.3. Instalación de ant**

En el CD adjunto se incluye un archivo comprimido con la versión de Ant. Sólo hay que descomprimirlo en una carpeta y luego añadir el directorio donde se encuentra el archivo ejecutable al Path para poder ejecutarlo desde línea de comando.

#### A.4. Instalación de Struts, JSTL y JDOM

En el CD adjunto se incluyen unos ficheros comprimidos que sólo necesitan de ser descomprimidos para instalarse.

#### A.5. Instalación de JBoss

En el CD adjunto se incluye un instalador de fácil manejo.

#### A.6. Instalación de la aplicación

Una vez instalados todos los programas necesarios para la correcta ejecución de la aplicación web, se procederá a la instalación de la misma.

Para ello primero ejecutaremos el script instalador de la base de datos que se incluye en el CD adjunto (*PostgreSQLCreateTables.sql*) en una consola de PostgreSQL. La ejecución de este script puede llevar varios minutos.

Se procederá a continuación a copiar dentro del directorio de JBoss *server/default/deploy/* los siguientes archivos que se incluyen en el CD adjunto: *postgres-ds.xml* y *EJBMapView.ear*.

Finalmente sólo restaría arrancar el servidor JBoss para hacer funcionar la aplicación, que estaría en *http://localhost:8080/Map Viewer*.

Si se quisiera ejecutar desde Eclipse, tendríamos que añadirlo como proyecto Java, modificar los ficheros *CommonProperties.xml* y *ConfigurationParameters.properties* con los paths de los directorios donde se encuentran los recursos utilizados en el desarrollo de la aplicación (tales como JBoss, Struts, etc.) y lanzar la tarea Ant *deployejbear* para compilar el proyecto y la tarea

Ant *initdb* para inicializar la base de datos.



## Contenido del CD adjunto

Con la presente memoria se adjunta un CD-ROM con los siguientes contenidos:

1. Esta documentación en formato PDF para su cómoda visualización, así como los ficheros utilizados para su creación con L<sup>A</sup>T<sub>E</sub>X, todo ello dentro de la carpeta *Memoria*.
2. Todos los componentes software necesarios para instalar (y compilar en caso de que se quisiese) la aplicación: Java JDK, JBoss, PostgreSQL, JDOM, Jakarta Struts, JSTL, todo ello dentro de la carpeta *Programas*.
3. Los ficheros necesarios para ejecutar la aplicación sin instalar: el fichero .ear de la aplicación, los scripts sql para la creación y borrado de tablas en la base de datos (PostgreSQLCreateTables.sql y PostgreSQLDeleteTables.sql), y el fichero postgres-ds.xml que se debe copiar en el directorio de JBoss *server/default/deploy/* al igual que el fichero .ear.
4. El código fuente de la aplicación dentro de la carpeta *MapViewer-src-1.0* y los diagramas UML en formato MagicDraw, dentro de la subcarpeta *Diagramas*.





## Uso de código licenciado

Para la realización de este proyecto se han utilizado una serie de librerías “open source” que se encuentran bajo la licencia que se muestra a continuación:

*Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:*

- *Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.*
- *Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.*
- *Neither the name of the organization nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.*

*THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS “AS IS” AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WA-*

*ARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.*

## C.1. J2EE Examples

Los ficheros *StandarUtil.jar* y *WebUtil.jar* empleados en este proyecto forman parte de los ejemplos *J2EE-Examples* del profesor Fernando Bellas Permu. Estos ejemplos se emplean en la asignatura de Integración de Sistemas que se imparte en la Facultad de Informática de la Universidade da Coruña.

## C.2. Javascript Image Cropper

Estas librerías Javascript se usan para hacer la selección de una zona geográfica en un mapa.

### C.3. WMSMap

Estas librerías Javascript se usan para facilitar el manejo de los mapas y las diversas peticiones a los WMSs.

### C.4. TinyMCE

Este editor HTML en Javascript se usa para facilitar la introducción del contenido HTML de los POI (Points Of Interest, Puntos de interés).





## Glosario de acrónimos

**AJAX** Asynchronous JavaScript And XML

**API** Application Program Interface

**BD** Base de Datos

**CASE** Computer Aided Software Engineering

**CSS** Cascading Style Sheets

**DOM** Document Object Model

**EJB** Enterprise JavaBean

**EJB-QL** EJB Query Language

**GIF** Graphics Interchange Format

**HTML** HyperText Markup Language

**HTTP** HyperText Transmission Protocol

**IDE** Integrated Development Environment

**J2EE** Java 2 Enterprise Edition

**J2SE** Java 2 Standard Edition

**JDBC** Java Database Connectivity

**JNDI** Java Naming and Directory Interface

**JPEG (JPG)** Joint Photographic Experts Group

**JSP** Java Server Pages

**JSTL** Java Standard Tag Library

**MDB** Message-Driven Beans

**MVC** Model-View-Controller

**OGC** Open Geospatial Consortium

**OOSE** Object Oriented Software Engineering

**PDF** Portable Document Format

**PNG** Portable Network Graphics

**SFSB** Stateful Session Beans

**SGBD** Sistema Gestor de Base de Datos

**SIG** Sistema de Información Geográfica

**SLD** Styled Layer Descriptor

**SLSB** Stateless Session Beans

**TO** Transfer Object

**UML** Unified Modeling Language

**URL** Uniform Resource Locator

**W3C** World Wide Web Consortium

**WFS** Web Feature Service

**WMS** Web Map Service

**XHTML** eXtensible Hypertext Markup Language

**XPath** XML Path Language

**XML** Extensible Markup Language

# E

## Glosario de términos

**Aplicación web** Se trata de un “software” que los usuarios usan desde un servidor web a través de Internet o de una intranet. Últimamente, están obteniendo una creciente popularidad gracias, entre otras razones, a la habilidad para actualizarlas y mantenerlas sin distribuir e instalar software en miles de potenciales clientes.

**Autenticar** Demostrar, en la red, que su identidad es realmente la que dice ser.

**Caso de uso** Especificación de las secuencias de acciones, incluyendo secuencias, variantes y secuencias de error, que pueden ser efectuadas por un sistema, subsistema o clase por interacción con autores externos.

**Clase** Definición de un tipo de objetos que tienen unas características comunes. Una clase es un patrón para crear nuevos objetos. Una clase contiene tanto atributos como métodos.

**Componente** Una parte física reemplazable de un sistema que empaqueta su implementación y es conforme a un conjunto de interfaces a las que proporciona su realización.

**Cookie** Un fragmento de información que se almacena en el disco duro del

visitante de una página web a través de su navegador, a petición del servidor de la página. Esta información puede ser luego recuperada por el servidor en posteriores visitas.

**Driver** Los drivers o controladores, son los encargados de actuar como interfaz entre dos sistemas para permitir la comunicación entre ellos.

**Escalabilidad** Medida de la capacidad de un sistema para mejorar su rendimiento incorporando nuevos elementos de procesamiento sobre los que ejecutarse sin la necesidad de modificar su implementación para ello.

**Framework** Estructura de soporte sobre la cual puede construirse un producto. Conjunto o sistema de reglas, ideas o creencias usadas para planear o decidir.

**Hipertexto** Paradigma en la interfaz del usuario cuyo fin es el de presentar documentos que puedan, según la definición de Ted Nelson, “bifurcarse o ejecutarse cuando sea solicitado” (branch or perform on request).

**Interfaz** Dispositivo de comunicación entre dos sistemas. Conexión física y funcional entre dos entidades independientes.

**Java** Lenguaje de programación orientado a objetos desarrollado por James Gosling y sus compañeros de Sun Microsystems al inicio de la década de los 90. A diferencia de los lenguajes de programación convencionales, que generalmente están diseñados para ser compilados a código nativo, Java es compilado en un bytecode que es ejecutado por una máquina virtual Java.

**JavaScript** Lenguaje interpretado orientado a las páginas web, con una sintaxis semejante a la del lenguaje Java.

**Navegador web** Aplicación software que permite al usuario recuperar y vi-

sualizar documentos de hipertexto, comúnmente descritos en HTML, desde servidores web de todo el mundo a través de Internet.

**Objeto** Un objeto es una entidad que tiene un estado y un conjunto definido de operaciones (métodos) que operan sobre este estado. El estado se representa por un conjunto de atributos del objeto. Las operaciones asociadas con el objeto dan servicio a otros objetos (clientes) que piden estos servicios cuando se necesita alguna operación (por medio de mensajes).

**Paquete** Término que denota un mecanismo de propósito general para organizar en grupos los elementos.

**Patrón Arquitectónico** Patrón de alto nivel que propone una arquitectura global para una aplicación.

**Patrón de Diseño** Solución a un problema de diseño no trivial que es efectiva y reusable.

**Request** Cualquiera de las “órdenes” que se envían a un servidor por medio de una aplicación cliente.

**Requisito funcional** Requisito que especifica una acción que debe ser capaz de realizar el sistema, sin considerar restricciones físicas.

**Requisito no funcional** Requisito que especifica propiedades del sistema, como restricciones del entorno o de implementación, rendimiento, dependencias de la plataforma, mantenibilidad, extensibilidad o fiabilidad.

**Servlet** Clase Java que puede recibir peticiones, normalmente HTTP, y generar una salida, normalmente HTML, WML o XML.

**Subsistema** Componente importante de un sistema organizado de una forma coherente. Un sistema puede ser dividido en subsistemas usando particiones

o niveles.

**Tag** Etiqueta. Se prefiere el término anglosajón al hispano porque está muy extendido en la jerga informática, sobre todo en los aspectos referentes a los lenguajes de marcas como HTML o XML.

**Web** La World Wide Web (del inglés, Telaraña Mundial), la web o WWW, es un sistema de hipertexto que funciona sobre Internet.

## Bibliografía

- [1] Fernando Bellas Permuy. Apuntes de la asignatura integración de sistemas, 2006. <http://www.tic.udc.es/~fbellas/teaching/is/index.html>.
- [2] Miguel Ángel Rodríguez Luaces. Apuntes de la asignatura bases de datos 3, 2006.
- [3] E. Gamma, R. Helm, R. Johnson, and J. Vlissides. *Patrones de Diseño*. Addison-Wesley, 2003.
- [4] J. Falkner, K. Jones, and et al. *JavaServer Pages: The J2EE Technology Web Tier*. Addison-Wesley, 2003.
- [5] J. Rumbaugh, I. Jacobson, and G. Booch. *The Unified Modeling Language*. Addison-Wesley, 1999.
- [6] Javascript image cropper. Página web. <http://www.defusion.org.uk/code/javascript-image-cropper-ui-using-prototype-scriptaculous/>.
- [7] Wms javascript library. Página web. <http://wms-map.sourceforge.net/>.

- [8] Tinymce. Página web. <http://tinymce.moxiecode.com/>.
- [9] Ajax. Página web. <http://www.uberbin.net/archivos/internet/ajax-un-nuevo-acercamiento-a-aplicaciones-web.php>.
- [10] Programación en javascript. desarrollo web. Página web. <http://www.desarrolloweb.com/manuales/26/>.
- [11] Apache software foundation. Página web. <http://struts.apache.org/>.
- [12] Java technology. sun microsystems. Página web. <http://www.java.sun.com/>.
- [13] Jboss. Página web. <http://www.jboss.org/>.
- [14] Junit. Página web. <http://www.junit.org/>.