Universidad Nacional Autónoma de Nicaragua.

UNAN – León.

MAESTRÍA EN COMPUTACIÓN CON ÉNFASIS EN GESTIÓN DE LA INFORMACIÓN.

2006 – 2008.

Desarrollo de una Aplicación Web con ASP .NET 2.0, basada en un modelo de tres capas, a implementarse en el componente curricular Programación Orientada a la Web II, de las titulaciones Ingeniería en Sistemas de Información e Ingeniería Telemática, Dpto. Computación UNAN – León.

Autor:

Ing. Valeria Medina Rodríguez.

Tutor:

Dr. Francisco Javier Ceballos Sierra.

León, Julio 2008

A Dios, por haberme iluminado y guiado por el gozoso sendero de la sabiduría.

A mi familia por ser el motor que impulsa mi vida, por su apoyo incondicional, simplemente gracias por ser la mejor familia del mundo.

A mi tutor Francisco Javier Ceballos, fuente de inspiración y de sabiduría, gracias por compartir conmigo sus conocimientos, por sus consejos, y por haber confiado en mí.

A todos los que de una u otra forma me han acompañado a lo largo de estos dos años, y colaboraron en el desarrollo de este proyecto.

ÍNDICE

Desarrollo de una Aplicación Web con ASP .NET 2.0, basada en un modelo de tres capas, a implementarse en el componente curricular Programación Orientada a la Web II, de las titulaciones Ingeniería en Sistemas de Información e Ingeniería Telemática, Dpto. Computación UNAN – León.

CAPÍTULO 1: ASPECTOS INTRODUCTORIOS.

1. INTRODUCCIÓN

Durante los últimos años el mundo de las nuevas tecnologías ha experimentado un gran auge, lo que ha propiciado una revolución en el ámbito de la sociedad de la información.

Hoy en día, resulta habitual conectarse a Internet para mantenernos informados acerca de cualquier cosa que sea de nuestro interés. Sin embargo, cada vez son más los que piensan que el modelo de comunicación concebido hasta ahora, basado en la figura del emisor-mensaje-receptor, ha quedado obsoleto, dando lugar a una nueva concepción de las comunicaciones en la que el usuario tiene algo más que aportar.

Nace con ello todo un mundo de posibilidades de interacción, con las nuevas tecnologías de la Información y Comunicación (TIC), que abarca desde la participación en foros y charlas, hasta la publicación de páginas Web y blogs.

De esta nueva realidad surge la idea de este proyecto, que se centra en la creación de un Portal Web de Noticias, y que será implementado en el componente curricular Programación Orientada a la Web II, de las titulaciones de Ingeniería en Sistemas de Información e Ingeniería Telemática, todo esto con objeto de armar al estudiante en el nuevo mundo de las TIC.

Ante todo este amplio abanico de posibilidades y servicios a disposición del usuario, este Portal de Noticias, podrá servir como modelo de referencia para el diseño de cualquier aplicación dinámica de este tipo.

2. ANTECEDENTES

La presente tesis tiene como antecedente dos trabajos monográficos, el primero un **Portal de noticias en ASP .Net AJAX**, realizado por Cristina Fumanal y Eduardo Barea Ropero, dicho portal fue desarrollado bajo la plataforma .NET y tiene como temática la gestión de noticias de carácter musical y el segundo **Mono en aplicaciones Web**, desarrollado por Hugo Párraga Martín, basado en aplicación ASP .NET que crea portales de noticias y cuyo objetivo fundamental fue demostrar la compatibilidad de las plataformas Mono sobre Linux y Microsoft .NET., ambos dirigido por el profesor Francisco Javier Ceballos Sierra, en el año 2007, en la Universidad de Alcalá – España, Escuela Politécnica Superior, Departamento de Automática

Tomando como punto de referencia los portales antes citado, se programará un Portal de Noticias, adaptándolo a las necesidades educativas de los estudiantes y las exigencias de las empresas en el ámbito de las aplicaciones para Internet.

Si bien es cierto, ya se han citado los antecedentes del proyecto a desarrollar, en el entorno a ser aplicado constituye una novedad tecnológica, pues actualmente la labor educativa en el área de programación Web, no contempla el desarrollo de aplicaciones completas como la que aquí se desarrolla.

3. JUSTIFICACIÓN

Dado al auge que actualmente han cobrado las comunicaciones y más específicamente la comunicación mediante aplicaciones en Internet, se considera necesario dotar al estudiante con herramientas que respondan a las exigencias del mercado laboral.

En nuestro país, son cada vez más las empresas e instituciones que se suman a la implementación de lenguajes de alto nivel y más específicamente leguajes orientados a la Web .NET, en este sentido es necesario que el egresado de Ingeniería en Sistemas e Ingeniería Telemática cuente con un amplio abanico de conocimientos y prácticas en esta área.

Con el desarrollo de esta aplicación ASP .NET, basada en un modelo de tres capas, se guiará al estudiante paso a paso por el desarrollo de la misma, en este proceso se irán adquiriendo todos los conocimientos sobre gestión y diseño de un sitio Web.

Por otra parte, la aplicación Web a desarrollada, mejorará el proceso enseñanza - aprendizaje, y sobre todo preparará al estudiante en la inserción laboral del campo de la programación orientada a la Web, que actualmente representa una de las mayores demandas en el mercado laboral.

Este trabajo también contribuirá al alcance de los objetivos del departamento de Computación y al fortalecimiento de la labor docente del mismo, ya que basándose en el desarrollo paso a poso de esta aplicación, se diseñó el componente curricular Programación Orientada a la Web II.

4. OBJETIVOS

OBJETIVO GENERAL

Desarrollar una Aplicación Web con ASP .NET 2.0, basada en un modelo de tres capas, para implementarse en el componente curricular Programación Orientada la Web II, de las titulaciones Ingeniería en Sistemas de Información e Ingeniería Telemática, Dpto. Computación UNAN – León.

OBJETIVOS ESPECÍFICOS

1. Desarrollar una Aplicación Orientada a la Web con ASP .NET 2.0, empleando como lenguaje de programación C·#.

2. Elaborar un documento en el que se muestren los aspectos organizativos de la asignatura Programación Orientada a la Web II, tales como: situación, metodología, material didáctico, proceso evaluativo y temporización de la misma.

3. Desarrollar el componente teórico – práctico de Programación Orientada a la Web II, basado en la Aplicación Web, aquí desarrollada.

5. ESTRUCTUTA DEL LIBRO

La estructura del libro es la siguiente:

- Una primera parte, a modo de introducción, donde se tratarán aspectos sobre los objetivos que se persiguen con este trabajo, el por qué del trabajo, los antecedentes del mismo, entre otras cosas.

- Una segunda parte, en la que abordarán aspectos relacionados con la estructuración del componente curricular, evaluación, relación con otras asignaturas, metodología a emplear, así como la temporización de todo el contenido teórico y práctico a abordar.

- Una tercera parte a tratar, en la cual se desarrollarán los contenidos teóricos del componente curricular, pero de una forma resumida, citando los fuentes en donde el estudiante puede ampliar sus conocimientos, esto con el objeto de fomentar el auto – aprendizaje.

- Una cuarta parte en la que se hablará en profundidad de la arquitectura del sitio Web, su desarrollo, detalles de diseño y creación, y en definitiva, de todo el proceso básico de elaboración del portal, en este apartado se incluirá también la especificación de requisitos.

- Una quinta parte en forma de anexos, que incluirá aspectos abordados en todo el libro pero que se deseen ampliar.

CAPÍTULO 2: PROGRAMACIÓN ORIENTADA A LA WEB II.

1. INTRODUCCIÓN

En el año 1994 se crea, bajo el Proyecto de Colaboración de la UA (Universidad de Alcalá) y la UNAN – León (Universidad Nacional Autónoma de Nicaragua – León), un convenio entre el Departamento de Automática de la Universidad de Alcalá de Henares (UAH) y el Departamento de Computación de la Universidad Autónoma de Nicaragua (UNAN-León) con el objetivo de apoyar la carrera de Licenciatura en Computación ahora Ingeniería en Sistemas de Información ofertada en la UNAN-León y posteriormente la Ingeniería Telemática. El convenio está dirigido a impulsar la excelencia en la enseñanza que permitirá formar mejores profesionales en el área de la informática.

Desde su creación ambas carreras han pasado por numerosas transformas curriculares con el afán de mejorar la formación académica de los estudiantes, la más significativa de estas transformas es la que se vive actualmente: del sistema de bloque al sistema de crédito.

El nacimiento de este nuevo sistema ha hecho que el Pénsul Académico de las carreras, sea ajustado a las nuevas tecnologías del mercado, con el fin de dotar al estudiante de mejores herramientas, que le permitan ser competitivo en el campo laborar, de esto han nacido nuevos componentes curriculares, y entre ellos el componente que aquí nos ocupa Programación Orientada a la Web II.

Se presenta la situación actual de la asignatura dentro del Pénsul Académico de Ingeniería en Sistemas e Ingeniería Telemática, su relación con otras asignaturas, la metodología y material didáctico para impartirla, así como la planificación del contenido teórico y práctico de la asignatura y la bibliografía recomendada.

Después del haber establecido todo lo relacionado con el componente curricular, se abordarán dos grandes partes. En la primera se desarrollan los contenidos teóricos que son necesarios para cumplir los objetivos de la asignatura y son los fundamentos básicos para la comprensión de las prácticas. En la segunda parte se desarrollan las prácticas de laboratorio que permitirán a los estudiantes reforzar y aplicar los conocimientos adquiridos en las clases teóricas y desarrollar el sitio Web aquí propuesto.

2. OBJETIVOS DE LA ASIGNATURA

1. Proporcionar al estudiante los conocimientos teóricos relacionados con el desarrollo de aplicaciones Web, basadas en ASP .NET.

2. Dirigir paso a paso al estudiante en el desarrollo básico de la aplicación Web: Portal de Noticias.

3. Fomentar el auto-aprendizaje en los estudiantes.

3. SITUACIÓN DE LA ASIGNATURA

La asignatura Programación Orientada a la Web II es de carácter obligatorio, se ha de impartir en las carreras de Ingeniería en Sistemas de Información e Ingeniería Telemática, de la UNAN-León, ofertada por el departamento de computación de la Facultad de Ciencias y Tecnología de dicha Universidad.

Las carreras se desarrollan a lo largo de 10 semestres de estudios (5 años académicos/lectivos). Esta asignatura se impartirá en el VIII semestre del sistema de crédito actual.

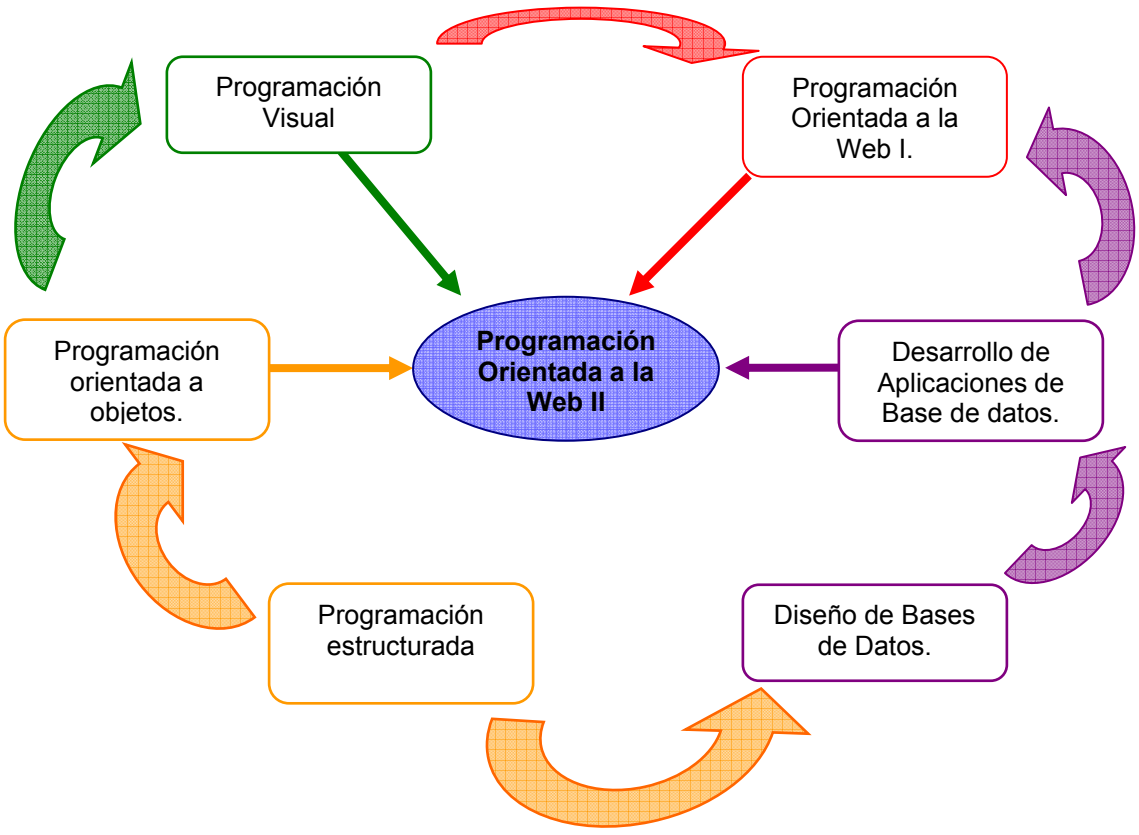
La asignatura consta con un total de 96 horas las cuales incluyen tanto tiempo de teoría como de laboratorio distribuidas a los largo de 16 semanas lectivas. A continuación se presenta la división de las horas entre la teoría y la práctica:

	Horas Semanales	Horas Semestrales
Teoría	2	32
Práctica	4	64
Un total de 96 horas al Semestre		

4. RELACIÓN CON OTRAS ASIGNATURAS

La formación académica no consiste en el aprendizaje de áreas aisladas del conocimiento, sino que es un proceso que consta de un conjunto de conocimientos que están estrechamente relacionados, y que una vez integrados le permiten al alumno, desarrollar habilidades en el campo profesional.

Es este contexto, no se puede realizar la planificación de este componente curricular, sin tomar en cuenta la relación que la asignatura Programación Orientada a la Web II, tiene con otras asignaturas impartidas al alumno en los años anteriores, y que sirven de base para el correcto entendimiento de la asignatura que estamos abordando. Por eso se hace importante plasmar la relación que tiene la asignatura de Programación Orientada a la Web II con otras asignaturas, relación que se muestra primero de manera gráfica y que se detalla más adelante.



En la figura anterior se ha dividido en 4 grupos las asignaturas con las cuales se relaciona la Programación Orientada a la Web II, representado cada grupo con un color diferente.

Fundamentos de programación:

- *Programación estructurada:* Se imparte en el III semestre, que se corresponde con el I semestre del II año de la carrera, aquí se sentarán las bases de la programación, de esta base dependerá la asimilación por parte del estudiante de todo el paradigma que incluye la programación.

- *Programación orientada a objetos*: Se imparte en el IV semestre, que se corresponde con el II semestre del II año de la carrera, donde se estudiarán todos los aspectos relacionado con clases, clases derivadas, excepciones entre otras cosas, conceptos que serán utilizados en el desarrollo de la aplicación Web.

Bases de datos.

- *Diseño de base de datos:* Se imparte en el V semestre, que se corresponde con el I semestre de III año, aquí el estudiante obtendrá los conocimientos básicos sobre bases de datos, relaciones y el lenguaje de consulta SQL, importantes para el desarrollo de la aplicación Web.

- *Desarrollo de aplicaciones de base de datos:* Se imparte en el VI semestre, que se corresponde con el II semestre de III año, el estudiante aprenderá a estructurar aplicaciones más complejas, empleando bases de datos y un entorno visual.

Programación Visual

- *Programación Visual:* Se imparte en el VI semestre, que se corresponde con el II semestre de III año, es aquí donde el estudiante aprende la estructura básica de una aplicación con entorno gráfico, todo esto en lenguaje de programación C#, bajo el cual está diseñado el componente curricular que aquí nos ocupa. Con respecto a la carrera de Ingeniería en Sistema, habrá una ampliación de este componente, con *Programación Visual II:* que se imparte en el VII semestre, que se corresponde con el I semestre del IV

año, donde el estudiante aprenderá conocimientos más avanzados sobre el diseño de interfaces gráficas, trabajará también con formularios y servicios Web, esta asignatura constituye la base fundamental sobre la cual se apoyará el desarrollo de Programación Orientada a la Web II.

Programación orientada a la Web

- *Programación orientada a la Web I:* Se imparte en el VII semestre, que se corresponde con el I semestre del IV año, aquí el estudiante tendrá una mayor aproximación, sobre la programación en Internet, diseñando formularios que interactúen con bases de datos remotas, todo esto basándose en el lenguaje HTML y JavaScript, aspectos muy importante para tener éxito en el desarrollo de la aplicación Web, que se propone.

5. METODOLOGÍA Y MATERIAL DIDÁCTICO

Debido al impacto que tendrá este curso en los estudiantes de ambas, sea concebido para desarrollarse con un alto sentido práctico, lo que permitirá que el estudiante en el aula de clases aplique todos los conocimientos de programación que se le han impartido.

Metodología didáctica:

- *En la parte Teórica:*

La metodología a utilizar para impartir este componente curricular serán las lecciones magistrales con una duración de dos horas (1 vez a la semana), durante las cuales se abordará de forma planificada el contenido teórico del componente curricular.

Debido al carácter práctico del componente, las lecciones serán complementadas con ejemplos que ayuden a una mejor comprensión de los conceptos expuestos. Además se propondrá la realización de trabajos extra clases tanto en la parte teórica como en la práctica para que el estudiante logre finalizar y personalizar el sito Web, despertando de esa manera la capacidad de búsqueda y el auto - aprendizaje en los alumnos.

- *En la práctica:*

Las prácticas de laboratorio, se llevarán a cabo en dos sesiones semanales, con duración de 2 horas cada sesión, se realizará una explicación de la temática ha bordar en cada sesión y puesto que las prácticas están guiadas, el profesor sólo aclarará dudas o resolverá algún otro problema. Se propondrán extensiones a las prácticas con el objetivo de incentivar al alumno a profundizar en el desarrollo de la aplicación Web.

Material didáctico:

Tanto para la parte práctica como para la parte teórica se utilizará el mismo material didáctico que será:

- La pizarra para las explicaciones y ejemplos en que sean necesarios.
- Láminas de acetato y retro-proyector.
- El portal de Noticias, sobre el cual se basa este componente curricular, esto con el objeto de que el alumno, compare, verifique, personalice y mejore el portal que él desarrollará a lo largo del curso.
- Sitio Web (Portal de Noticias), donde se publicarán enlaces de interés, noticias relacionadas con la Programación Orientada a la Web, y manuales de referencias considerados de importancia para el abordaje de la asignatura,
- Acceso a la bibliografía básica para esta asignatura y a la información impresa del material de la Aplicación Web, la cual estará disponible en la biblioteca del departamento.

6. PLANIFICACIÓN TEMPORAL

Para realizar una planificación objetiva del componente curricular, se debe tomar en cuenta la cantidad de horas disponibles para la parte teórica y la práctica.

Partiendo del calendario académico del año lectivo 2008 aprobado en la UNAN – León y tomando en cuenta que la asignatura será impartida durante el II semestre del año, obtenemos un total de 16 semanas, pero considerando los días festivos y dando margen a otras incidencias que puedan ocurrir durante el semestre, podemos deducir un total de 14 semanas para cumplir con el contenido de la materia.

El número de horas asignadas a cada tópico se ha calculado en base a la profundidad con que se abordará cada tema y el propio contenido del mismo.

De tal manera que la planificación temporal, como resultado de los elementos planteados anteriormente, se expone en el cuadro que se presenta a continuación:

Panificación temporal de la parte teórica:

Nº de Tema	Tema	Horas
0	Presentación de la Asignatura	1
1	Seguridad de Aplicaciones ASP .NET	6
2	Páginas Maestras	2
3	Temas y Máscaras de ASP .NET 2.0	4
4	Personalización de Sitos Web: Perfiles	2
5	Administración de Miembros y Funciones	4
6	RSS	4
7	AJAX y ASP .NET AJAX	5
Total de horas de Teoría		28

Para el cálculo de las horas de laboratorio, se debe tomar en cuenta, que las prácticas están guiadas y que no se está obligado a trabajar únicamente en el laboratorio, pues estás prácticas no necesitan ninguna infraestructura especial.

Planificación temporal de la parte práctica:

Nº de Práctica	Práctica	Horas
1	Introducción al Portal Web: Arquitectura del Proyecto: Requisitos Software	2
2	Modelo de Datos	4
3	Objetos de Negocio	4
4	Capa de Acceso a Datos (DAL)	8
5	Capa de Lógica de Negocio (BLL)	4
6	Capa de Presentación	22
7	Modo Administrador	8
8	Personalización del Sitio Web	4
Total de horas de Laboratorio		**56**

7. METODOLOGÍA DE EVALUACIÓN

La asignatura está compuesta por dos partes, una teórica y otra práctica, teniendo esta última la mayor carga en cuanto a contenido y por tanto en cuanto a porcentaje evaluativo.

A lo largo del semestre se realizarán 3 evaluaciones parciales, cuyo promedio representará el 100% de la nota Final, en este sentido las tres evaluaciones tienen igual ponderación.

Las tres evaluaciones parciales se desglosan de la siguiente manera:

- Seguimiento de las prácticas…… 40%
- Examen Teórico…… 40%
- Examen Práctico…….. 20%
- **Total …… 100%**

Seguimiento de las prácticas:
El 40% correspondiente al seguimiento de las prácticas, se obtendrá a través de la entrega en tiempo y forma de las prácticas de laboratorio, propuestas. El valor de cada práctica se distribuirá en función del número y la complejidad de las mismas. Debido a que la asignatura está autoguiada, el avance que tenga el alumno en cuanto al desarrollo del Portal y las innovaciones que él pueda incorporar, también serán consideradas dentro de esta ponderación, es decir, el trabajo independiente del estudiante será calificado.

Examen Teórico:
El examen teórico que corresponde al 40% de cada parcial constará de preguntas que permitan evaluar el grado de compresión de los conceptos básicos que tiene el estudiante. También contendrá preguntas de carácter analítico y ejercicios en los cuales el alumno deberá de demostrar la capacidad de aplicar los conocimientos adquiridos.

Examen Práctico:
Consistirá en el planteamiento de pequeños tópicos relacionados, con los abordados en la parte práctica y a los que el estudiante deberá darle una solución, esto corresponde al 20% de cada parcial.

8. CONTENIDO DEL TEMARIO

TEMA 0: PRESENTACIÓN DE LA ASIGNATURA.

1. **Objetivos de la asignatura.**
2. **Contenido de la asignatura.**
3. **Temporización.**
4. **Metodología de evaluación.**
5. **Bibliografía a utilizar.**
6. **Horarios de tutorías.**

TEMA 1: SEGURIDAD DE APLICACIONES ASP .NET

1. **Introducción**
2. **Arquitectura ASP.NET**
3. **Modos de autenticación**

    Autenticación de Windows
    Autenticación Passport
    Autenticación predeterminada o none
    Autenticación de formularios

4. **Modos de autorización**
5. **Configuración de seguridad de la aplicación**

    Configuración del sitio Web - archivo Web.config

6. **Flujo de datos de seguridad en ASP.NET**

    Escenario: Autenticación de formularios

7. **Controles de sesión de ASP.NET**

    Control Login (Inicio de Sesión)
    Control LoginView
    Control LoginStatus

Control LoginName

Control LoginStatus

Control CreateUserWizard

Control ChangePassword

8. **Herramientas de Administración de sitios Web**

Ficha Seguridad

Ficha Aplicación

Ficha Proveedor

Ficha Seguridad

TEMA 2: PÁGINAS MAESTRAS

1. **Descripción**
2. **Cómo trabajar con páginas maestras**
3. **Ventajas de las páginas maestras**
4. **Comportamiento en tiempo de ejecución**

TEMA 3: TEMAS Y MÁSCARAS DE ASP .NET 2.0

1. **Introducción**
2. **Máscaras de temas y controles**
3. **Ámbito de los temas**
4. **Prioridad en la configuración de temas**
5. **Propiedades definibles mediante temas**
6. **Temas frente a CSS**
7. **Consideraciones de seguridad**
8. **Cómo definir temas en ASP .NET**
9. **Cómo aplicar temas en ASP.NET**

TEMA 4: PERSONALIZACIÓN DE SITIOS WEB: PERFILES

1. **Introducción**
2. **Funcionamiento de los perfiles en ASP.NET**
3. **Identificación de usuario para las propiedades de perfil en ASP.NET**

TEMA 5: ADMINISTRACIÓN DE MIEMBROS Y FUNCIONES

1. **Introducción**
2. **Descripción**

TEMA 6: RSS

1. **¿Qué es RSS y para qué sirve?**
2. **¿Cómo se usa RSS?**
3. **¿Cómo crear RSS?**

TEMA 7: AJAX y ASP .NET AJAX

1. **AJAX**

    Introducción
    Tecnologías que conforman AJAX
    Ejecutar aplicaciones AJAX
    Ventajas e inconvenientes de AJAX
    AJAX contra Aplicaciones Web tradicionales – ¿cómo funciona?
    Limitaciones

2. **ASP .NET AJAX**

    Introducción
    Arquitectura de ASP .NET AJAX
    ASP.NET AJAX SERVER CONTROLS
    ASP .NET Control Toolkit

CAPÍTULO 3: ASPECTOS TEÓRICOS

1. SEGURIDAD DE APLICACIONES ASP .NET

1.1. Introducción

La seguridad de los sitios Web es una cuestión de importancia fundamental, además de compleja, a la hora de desarrollar un sitio Web. La protección de un sitio requiere la elaboración cuidadosa de un plan que se ajuste a las necesidades específicas de seguridad de nuestra aplicación.

Una parte importante de muchas aplicaciones Web radica en la capacidad de identificar usuarios y controlar el acceso a los recursos. En el caso que nos concierne, la aplicación verá personalizado dinámicamente su contenido en función de la identidad del solicitante. Por ejemplo, si quien intenta acceder a nuestra Web es un usuario previamente registrado, tendrá acceso a un mayor número de contenidos que aquel que accede de forma anónima. ASP.NET trabaja mano a mano con Microsoft .NET Framework y Servicios de Microsoft Internet Information Server (IIS) para proporcionarnos un modelo de aplicación Web segura.

1.2. Arquitectura ASP.NET

Toda aplicación Web sigue un esquema básico de seguridad en el que los diferentes elementos interactúan de un determinado modo para procurar alcanzar una aplicación segura.

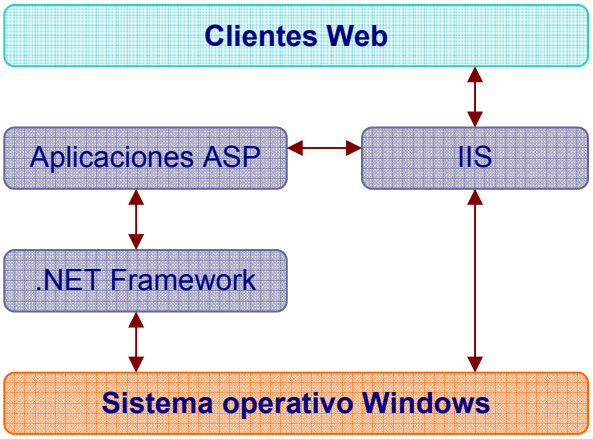


Fig. 1 Infraestructura ASP .NET en la seguridad.

Como se muestra en la **Fig. 1**, todos los clientes Web se comunican con las aplicaciones ASP.NET a través de Servicios de Microsoft Internet Information Server (IIS). IIS autentica la solicitud si fuera necesario y así estuviera configurado, y a continuación, pasa el control a la aplicación ASP.NET. Será esta quien decidirá si el cliente está autorizado o no para acceder al recurso solicitado.

Para ayudar a proteger la seguridad de una aplicación ASP.NET, se deben llevar a cabo las dos funciones principales que se describen a continuación.

Autenticación

Ayuda a comprobar que el usuario es precisamente quien dice ser. La aplicación obtiene las credenciales (diversas formas de identificación, como nombre y contraseña) de un usuario, y las valida consultando a una autoridad determinada. Si las credenciales son válidas, se considera a la entidad que ha enviado las credenciales como una entidad autenticada.

Autorización

La autorización es el proceso que verifica si el usuario autenticado tiene acceso a los recursos solicitados. Limita los derechos de acceso mediante la concesión o negación de permisos específicos a una identidad autenticada.

Flujo de seguridad con una solicitud

Los pasos siguientes esquematizan la secuencia de eventos cuando un cliente realiza una solicitud:

- Un cliente solicita una página .aspx que reside en un servidor IIS.
- Las credenciales del cliente pasan a IIS.
- IIS autentica al cliente y reenvía el símbolo (token) autenticado junto con la solicitud del cliente al proceso de trabajo de ASP.NET.
- Basándose en el símbolo autenticado procedente de IIS y la configuración de las aplicaciones Web, ASP.NET decide si debe suplantar a un usuario en el subproceso que está procesando la solicitud. A diferencia de lo que ocurre con Microsoft Active Server Pages (ASP), ASP.NET no suplanta de manera predeterminada al usuario autenticado.

Para habilitar la representación, hay que configurar el atributo impersonate de la sección de identidad en el archivo Web.config en true.

1.3. Modos de autenticación

A continuación se pasa a describir los distintos tipos de modos de autenticación existentes, haciendo especial hincapié en el último de ellos, el basado en formularios, que será sobre el que se fundamente la aplicación.

1.3.1. Autenticación de Windows

En la autenticación de Windows, IIS realiza la autenticación y el token autenticado se reenvía al proceso de trabajo de ASP.NET. La ventaja de utilizar la autenticación de Windows es que requiere una codificación mínima. Es idónea para suplantar la cuenta de usuario de Windows que IIS autentica antes de ceder la solicitud a ASP.NET. Con este modo lo que se hace es delegar en el sistema operativo la tarea de autenticación de usuarios, con lo que cual sólo podrán acceder ana aplicación ASP .NET aquellos usuarios que existen previamente en el sistema donde está instalada.

1.3.2. Autenticación Passport

La autenticación Passport es un servicio de autenticación centralizado proporcionado por Microsoft, que ofrece un registro único y servicios de perfil de núcleo para sitios miembros. Habitualmente, la autenticación Passport se utiliza cuando se necesita un registro único en varios dominios.

1.3.3. Autenticación predeterminada o none

La autenticación predeterminada se utiliza cuando no se requiere seguridad en una aplicación Web. Para este proveedor de seguridad es necesario un acceso anónimo. Entre todos los

proveedores de autenticación, la autenticación predeterminada proporciona el máximo rendimiento para la aplicación.

1.3.4. Autenticación de formularios

ASP.NET también proporciona un robusto servicio para aplicaciones que necesitan utilizar autenticación basada en formularios. La autenticación basada en formularios es un servicio de autenticación de ASP.NET que permite a las aplicaciones suministrar su propia interfaz de inicio de sesión y hacer su propia verificación de credenciales. ASP.NET permite autenticar usuarios y desviar a los usuarios no autenticados hacia la página de inicio de sesión. Este tipo de autenticación es una técnica habitual utilizada en muchos sitios Web.

La autenticación basada en formularios utiliza "cookies" permite autenticar a los usuarios, y permite a la aplicación realizar su propia verificación de credenciales.

La autenticación de formularios hace referencia a un sistema en el que la solicitudes no autenticadas se redirigen a un formulario de Lenguaje de marcado de hipertexto (HTML) en el que los usuarios escriben sus credenciales. Una vez que el usuario proporciona las credenciales y envía el formulario, la aplicación autentica la solicitud y el sistema emite un vale de autorización en el formulario de una cookie.

Esta cookie contiene las credenciales o una clave para readquirir la identidad. Las solicitudes subsiguientes del explorador automáticamente incluyen la cookie. Para que una aplicación pueda utilizar autenticación basada en formularios, se debe configurar <authentication> con la opción Forms en el archivo de configuración y denegar el acceso a los usuarios anónimos.

Los administradores usan autenticación basada en formularios para configurar el nombre de la "cookie" a usar, el tipo de protección, la dirección URL para la página de inicio de sesión, el tiempo de validez de la "cookie" y la ruta de acceso que se debe utilizar para la "cookie" suministrada.

Por último, es importante recalcar que los servicios de autenticación de ASP.NET dependen de los servicios de autenticación suministrados por IIS. Dependiendo del tipo de configuración que

hayamos otorgado a IIS podremos seleccionar uno u otro modo de autenticación en nuestra aplicación.

1.4. Modos de autorización

ASP.NET proporciona dos tipos de servicios de autorización:

- Comprobaciones de ACLs o permisos sobre un recurso para determinar si la cuenta de usuario autenticada puede obtener acceso a los recursos
- Autorización de URL, la cual autoriza una identidad para partes del espacio Web.

En nuestro caso nos decantaremos por una autorización de url, donde dependiendo del tipo de perfil de usuario que intente acceder a nuestra aplicación, se le concederá acceso a determinados recursos ubicados en distintas urls. Todo esto se puede hacer mediante las restricciones a carpetas en el sitio Web.

1.5. Configuración de seguridad de la aplicación

La configuración de seguridad de ASP.NET se define en los archivos Machine.config y Web.config. Como con la demás información de configuración, la configuración base y la predeterminada se establecen en el archivo Machine.config en el subdirectorio Config de la instalación .NET Framework actual. Podemos establecer una configuración específica del sitio y otra específica de la aplicación (incluidos los valores de reemplazo del archivo Machine.config) en los archivos Web.config en los directorios raíz del sitio Web y de la aplicación. Los subdirectorios heredan las configuraciones del directorio, a no ser que se reemplacen por un archivo Web.config del subdirectorio.

1.5.1. Configuración del sitio Web - archivo Web.config

Los valores de configuración del sitio Web se almacenan en un archivo XML denominado Web.config, que se encuentra en la carpeta raíz del sitio Web. La herramienta Administración de sitios Web, de la que hablaremos mas adelante, nos permitirá cambiar la configuración del sitio sin tener que editar manualmente el archivo Web.config.

La confirmación predeterminada de un sitio Web se hereda automáticamente de los archivos de configuración existentes para el equipo o para el servidor Web en su conjunto. Por ejemplo, el servidor Web podría tener una configuración predeterminada que se aplica a todos sus sitios. Mediante la herramienta Administración de sitios Web podemos crear y modificar valores de configuración del sitio Web que no sean heredados, así como reemplazar la configuración heredada en la medida que lo permita la configuración del sitio. Si los valores de configuración son heredados y no se pueden reemplazar, aparecen atenuados en la herramienta Administración de sitios Web, lo que indica que están deshabilitados.

Hay tres subsecciones principales en un archivo Web.config: las secciones autenticación, autorización e identidad. Los valores para cada elemento de seguridad normalmente se establecen en el archivo Machine.config y se reemplazan según sea necesario en el archivo Web.config en la aplicación. Todos los subdirectorios heredan automáticamente estos valores. No obstante, los subdirectorios pueden tener sus propios archivos de configuración que reemplazan valores heredados.

1.6. Flujo de datos de seguridad en ASP.NET

Existen varias maneras de diseñar la seguridad en las aplicaciones ASP.NET. A continuación pasaremos a describir el flujo de datos de seguridad en el escenario que se ajusta a las circunstancias de nuestra aplicación, basado en la autenticación de formularios mediante cookies.

1.6.1. Escenario: Autenticación de formularios

En el escenario de autenticación de formularios, la aplicación obtiene las credenciales, como el nombre y la contraseña, directamente del usuario y determina por sí misma su autenticidad. La aplicación no utiliza la autenticación de IIS, pero si la configuración de la autenticación de IIS puede afectar a la autenticación de formularios. Como norma, cuando se utiliza la autenticación de formularios, se habilita el acceso anónimo en IIS. Por otra parte, si los usuarios no pasan la autenticación de IIS, no pueden ponerse en contacto con la aplicación para proporcionar un nombre de usuario y una contraseña para la autenticación de formularios. El flujo de datos de este escenario se muestra en la **Fig. 2.**

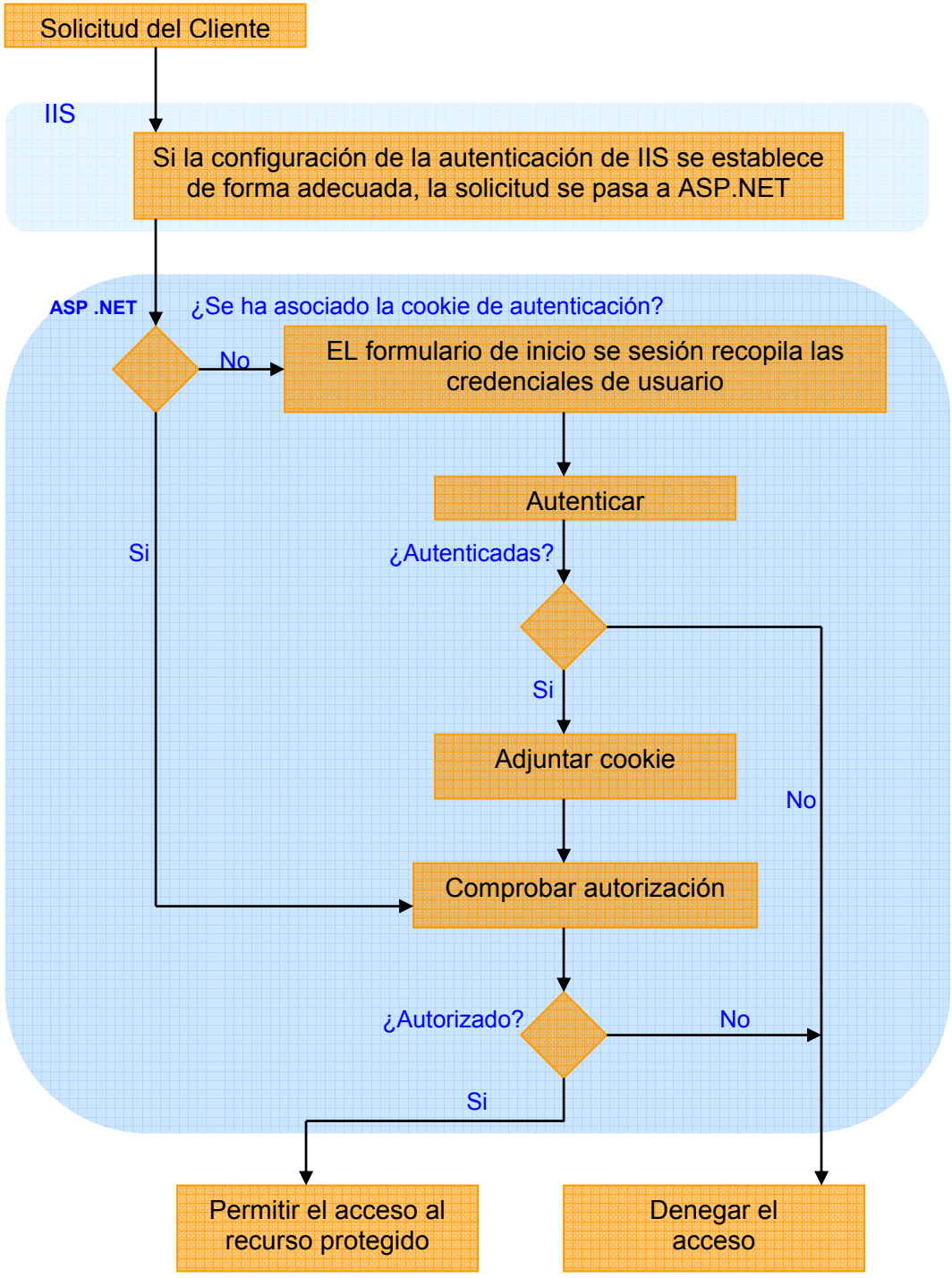


Fig. 2 Autenticación mediante formularios.

En la Fig. 2 se muestra la siguiente secuencia de eventos:

- Un usuario genera una solicitud de un recurso protegido.

- IIS recibe la solicitud y, puesto que el acceso anónimo de IIS está habilitado, IIS no realiza ninguna autenticación del usuario y la solicitud se pasa a la aplicación ASP.NET.
- Dado que el modo de autenticación de ASP.NET se ha establecido en formularios, la aplicación ASP.NET examina la solicitud para obtener un vale de autenticación de formularios (una cookie concreta). Si no hay ningún vale de autenticación asociado a la solicitud, ASP.NET redirige la solicitud a la página de inicio de sesión especificada en el archivo de configuración de la aplicación.
- En la página de inicio de sesión, el usuario escribe las credenciales requeridas, normalmente un nombre y una contraseña. El código de aplicación comprueba las credenciales para confirmar su autenticidad. Si se autentican las credenciales, el código de aplicación asocia a la respuesta un vale de autenticación que representa las credenciales del usuario. (No se incluye la contraseña.) Si se produce un error en la autenticación, la respuesta se devuelve con un mensaje de acceso denegado o se vuelve a mostrar el formulario de inicio de sesión. El vale de autenticación emitido se incluirá en las solicitudes que se realicen a la aplicación ASP.NET con posterioridad. ASP.NET examina la validez del vale mediante una comprobación de la autenticación de mensajes (MAC).
- Si se autentica al usuario, ASP.NET comprueba la autorización y puede conceder acceso al recurso solicitado inicialmente, redirigir la solicitud a alguna otra página o redirigirla a un módulo de autorización personalizado, donde se comprueba si las credenciales están autorizadas a tener acceso al recurso protegido. Si se produce un error de autorización, ASP.NET redirige al usuario a la página de inicio de sesión. Si se autoriza al usuario, se concede el acceso al recurso protegido la aplicación puede requerir una prueba adicional de las credenciales antes de autorizar el acceso al recurso protegido, dependiendo del diseño de la aplicación.

1.7. Controles de sesión de ASP.NET

Los controles de inicio de sesión de ASP.NET funcionan juntos para proporcionar una solución de inicio de sesión completa y sólida para las aplicaciones Web ASP.NET que no requiere programación. De forma predeterminada, los controles de inicio de sesión se integran con la suscripción a ASP.NET para ayudar a automatizar la autenticación del usuario en el sitio Web

mediante formularios. De manera predeterminada, los controles de inicio de sesión de ASP.NET se transmiten en forma de texto sin cifrar en HTTP.

Visual Studio provee varios controles que encapsulan la funcionalidad para implementar la seguridad proporcionada por la autenticación y la autorización: **Login, LoginView y LoginState,** entre otros.

1.7.1. Control Login (Inicio de Sesión)

El control Login muestra una interfaz de usuario para la autenticación de usuario. El control Login contiene cuadros de texto para el nombre de usuario y la contraseña, y una casilla de verificación que permite a los usuarios indicar si quieren que el servidor almacene su identidad utilizando la suscripción de ASP.NET y que los autentique automáticamente la próxima vez que visiten el sitio.

El control Login tiene propiedades para una presentación personalizada, para mensajes personalizados y para vínculos a otras páginas en las que los usuarios pueden cambiar su contraseña o recuperarla si la han olvidado.

1.7.2. Control LoginView

Asimismo, si se necesita visualizar diferentes contenidos para diferentes usuarios, basados en si el usuario está o no autenticado, y si lo está, qué funciones del sitio Web tiene, se puede hacer de una forma sencilla a través del control LoginView. Este control puede presentar dos vistas (plantillas):

- ***AnonymousTemplate:*** esta plantilla será visualizada cuando el usuario que intenta acceder al sitio no está autenticado.
- ***LoggedInTemplate:*** esta plantilla será visualizada cuando el usuario que intenta acceder al sitio Web ya está autenticado.

1.7.3. Control LoginStatus

El control LoginStatus muestra un vínculo de inicio de sesión para los usuarios que no están autenticados y un vínculo de cierre de sesión para los que están autenticados. El vínculo de inicio de sesión lleva al usuario a una página de inicio de sesión. El vínculo de cierre de sesión restablece la identidad del usuario actual para que sea un usuario anónimo.

1.7.4. Control LoginName

El control LoginName muestra el nombre de inicio de sesión de un usuario si el usuario ha iniciado la sesión mediante la suscripción de ASP.NET.

1.7.5. Control LoginStatus

El control PasswordRecovery permite recuperar las contraseñas del usuario basándose en la dirección de correo electrónico que se utilizó cuando se creó la cuenta. El control PasswordRecovery envía un mensaje de correo electrónico con la contraseña al usuario.

Podemos configurar la suscripción de ASP.NET para almacenar contraseñas mediante el cifrado no reversible. En este caso, el control PasswordRecovery genera una nueva contraseña en lugar de enviar la contraseña original al usuario.

También podemos configurar la suscripción para que incluya una pregunta de seguridad que el usuario debe contestar para recuperar una contraseña. Si lo hace, el control PasswordRecovery hace la pregunta y comprueba la respuesta antes de recuperar la contraseña.

El control PasswordRecovery requiere que la aplicación pueda reenviar el mensaje de correo electrónico a un servidor de Protocolo simple de transferencia de correo (SMTP). Podemos personalizar el texto y formato del mensaje de correo electrónico enviados al usuario estableciendo la propiedad MailDefinition. La información de la contraseña que se envía en un mensaje de correo electrónico se hace como texto no cifrado.

1.7.6. Control CreateUserWizard

El control CreateUserWizard recoge información de los posibles usuarios. De forma predeterminada, el control CreateUserWizard agrega el nuevo usuario al sistema de suscripciones de ASP.NET.

El control CreateUserWizard recopila la siguiente información sobre el usuario:

- ⇒ Nombre de usuario
- ⇒ Contraseña
- ⇒ Confirmación de contraseña
- ⇒ Dirección de correo electrónico
- ⇒ Pregunta de seguridad
- ⇒ Respuesta de seguridad.

Esta información se utiliza para autenticar a los usuarios y recuperar las contraseñas del usuario, si fuera necesario.

1.7.7. Control ChangePassword

El control ChangePassword permite a los usuarios cambiar su contraseña. El usuario debe proporcionar primero la contraseña original y, a continuación, crear y confirmar la nueva contraseña. Si la contraseña original es correcta, la contraseña del usuario se cambia a la nueva contraseña. El control también se encarga de enviar un mensaje de correo electrónico sobre la nueva contraseña.

El control ChangePassword incluye dos vistas con plantilla que se muestran al usuario. La primera es ChangePasswordTemplate, que muestra la interfaz de usuario que se utiliza para recopilar los datos necesarios para cambiar la contraseña del usuario. La segunda plantilla es SuccessTemplate, que define la interfaz de usuario que se muestra después de haber cambiado correctamente la contraseña del usuario.

El control ChangePassword funciona con usuarios autenticados y no autenticados. Si el usuario no se ha autenticado, el control solicita al usuario que escriba un nombre de inicio de sesión. Si

el usuario se ha autenticado, el control rellena el cuadro de texto con el nombre de inicio de sesión del usuario.

1.8. Herramientas de Administración de sitios Web

Visual Studio incorpora una nueva herramienta para gestionar los diversos aspectos de nuestra aplicación Web: la herramienta de administración de sitios Web de ASP.NET.

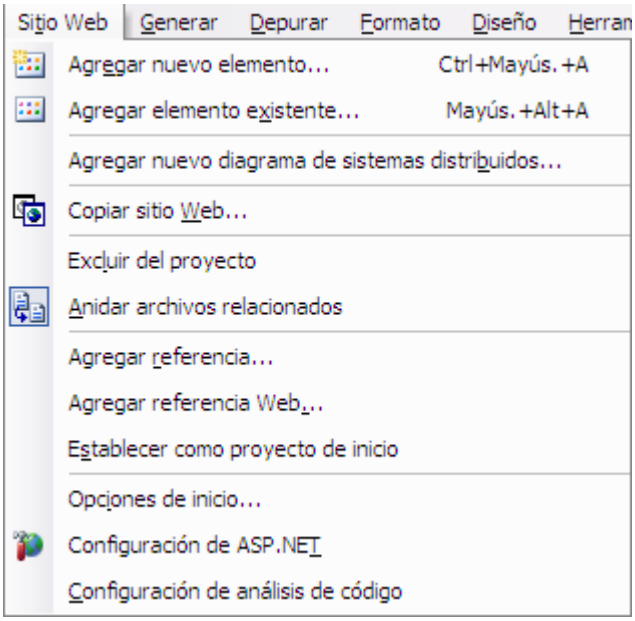
Con ella configuraremos todos los aspectos de nuestra aplicación relacionados con la seguridad: desde los proveedores que emplearemos, hasta la administración de los permisos de acceso a las carpetas. También gestionaremos la pertenencia a roles, y la administración de usuarios, pudiendo incluso llegar a definirlos desde este asistente.

La primera vez que se utiliza la herramienta Administración de sitios Web para administrar un sitio determinado, la herramienta crea un archivo Web.config si aún no existe.

De forma predeterminada, la herramienta Administración de sitios Web también crea una base de datos SQL Server en la carpeta App_Data del sitio Web para almacenar todos los datos relacionados con los usuarios, roles…etc. La base de datos tendrá una estructura predefinida y contará con una amplia colección de procedimientos almacenados para facilitar su gestión.

Para la mayoría de los valores de configuración, los cambios que se realizan con la herramienta Administración de sitios Web surten efecto de inmediato y se reflejan en el archivo Web.config.

Para acceder a esta herramienta, seleccionamos del menú de Sitio Web la opción Configuración de ASP.NET. También podemos hacerlo a través del menú de tareas de cualquiera de los controles de Inicio de Sesión.



La herramienta Administración de sitios Web presenta una interfaz con fichas en la que se agrupan las opciones de configuración relacionadas. Las fichas y las opciones de configuración que las fichas administran se describen en las secciones siguientes.

1.8.1. Ficha Seguridad

La ficha seguridad nos servirá para administrar las reglas de acceso, para ayudar a proteger determinados recursos dentro del Sitio Web y administrar cuentas de usuario y funciones.

Podremos especificar cómo se utiliza el sitio Web: a través de Internet (públicamente) o de una intranet (en una red de área local). Esto indica a su vez el tipo de modo de autenticación que se utilizará para el sitio Web. Los sitios Web de Internet utilizan el sistema de suscripciones ASP.NET, donde se definen cuentas de usuario de forma individual. ASP.NET utiliza un sistema de seguridad para restringir el acceso a determinadas cuentas de usuario o las funciones a las que pertenecen las cuentas de usuario. Los sitios Web de intranet utilizan autenticación de Windows, de modo que la información de inicio de sesión de Windows permite identificar a los usuarios.

1.8.2. Ficha Aplicación

Utilizaremos la ficha aplicación para administrar diversas opciones de configuración relacionadas con el sitio Web, entre las que se incluyen las siguientes:

- ⇒ Configuración de la aplicación, que corresponde a pares de nombres y valores que desea almacenar de forma centralizada y a los que desea obtener acceso mediante código desde cualquier parte del sitio Web.
- ⇒ Configuración SMTP, que determina cómo envía el correo electrónico el sitio.
- ⇒ Configuración de depuración y seguimiento.
- ⇒ Valores de configuración con o sin conexión, que desconectan el sitio Web (lo apagan) para operaciones de mantenimiento o para conectar una nueva base de datos Microsoft SQL Server Standard Edition.

1.8.3. Ficha Proveedor

Podemos utilizar la ficha Proveedor si deseamos probar o asignar proveedores para la suscripción y la administración de funciones del sitio Web. Los proveedores de bases de datos son clases a las que se llama para almacenar datos de aplicación para una característica determinada. De forma predeterminada, la herramienta Administración de sitios Web configura y utiliza una base de datos Microsoft SQL Server Standard Edition local en la carpeta App_Data del sitio Web. En su lugar, podemos utilizar un proveedor diferente, como una base de datos de SQL Server remota, para almacenar las suscripciones y los datos de administración de funciones.

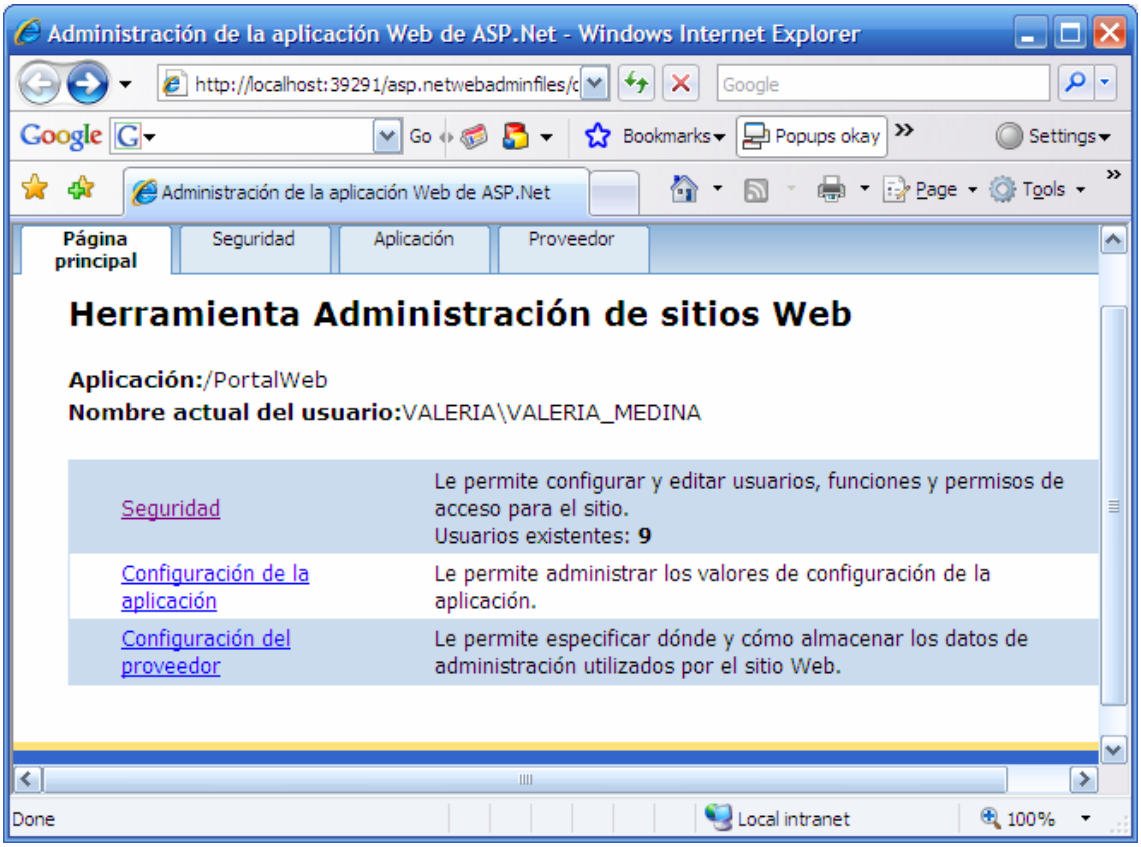


Fig. 3 Administración de la aplicación Web de ASP .NET

1.8.4. Configurando la Seguridad del Sitio Web

A continuación, comenzaremos a configurar la seguridad de la aplicación siguiendo las siguientes pautas.

Hacemos clic en Seguridad para poder configurar y editar usuarios, funciones y permisos de acceso para el sitio.

En la página principal de Seguridad, hacemos clic en Seleccionar Modo de Autenticación. En la siguiente página, seleccionamos Desde Internet, y a continuación hacemos clic en Listo. Esto modifica el tipo de autenticación en el archivo web.config de "Windows" a "Forms".

Acto seguido pasaremos a activar las llamadas funciones o roles, para así poder asignar determinados permisos y accesos dependiendo del tipo de peticionario que se dirija a nuestra Web.

En principio contaremos con dos roles. El primero de ellos, el de administrador, será el encargado de desempeñar labores de gestión de usuarios y contenidos del portal, funciones que serán especificadas en detalle mas adelante.

El segundo será el rol de moderador, cuyas atribuciones estarán únicamente relacionadas con la gestión del foro.

Lo primero que debemos hacer para lograr nuestro objetivo es hacer clic en la opción habilitar funciones, dentro del apartado Funciones. Una vez habilitado, podremos crear los roles que pretendemos,

A continuación estaremos en disposición de definir las reglas de acceso correspondientes a cada uno de los perfiles. Para ello acudimos a la opción Crear reglas de acceso, dentro del apartado Reglas de Acceso, en la página principal de seguridad, que nos abrirá una ventana como la mostrada en la **Fig. 4.**

En ella podremos decidir que carpetas (y con ellas, los contenidos subyacentes) estarán disponibles o no para cada tipo de perfil. Bastará con seleccionar que carpeta queremos que forme parte de la regla, a que función o rol queremos aplicarla, y que permiso le queremos aplicar. También podemos aplicar dichos permisos a colectivos genéricos, marcando las opciones Todos los usuarios o Usuarios anónimos, o a usuarios concretos, seleccionando la opción usuario y buscando un usuario concreto.

Para poder ver y administrar las reglas de acceso que acabamos de crear, acudimos a la opción Administrar Reglas de Acceso dentro del apartado Reglas de acceso.

Las reglas se aplican en orden. Se aplica la primera regla que coincide y los permisos de cada regla reemplazan los permisos de las siguientes. Esta página nos permite mover reglas hacia arriba o abajo, de forma que si las introducimos en el orden erróneo siempre podemos efectuar cambios. Asimismo podemos eliminarlas, y también dar lugar a nuevas reglas.

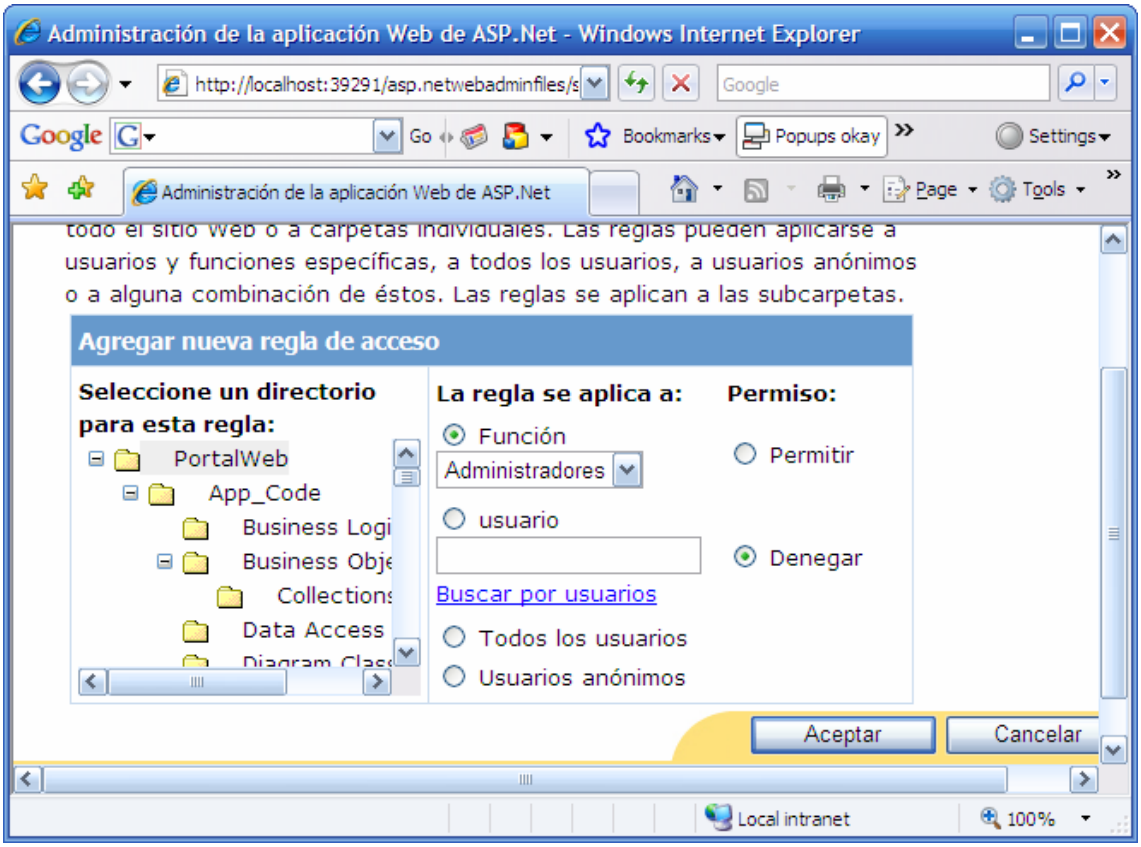


Fig. 4 Aplicar regla se acceso.

En cuanto a la gestión de usuarios, podemos crearlos directamente desde aquí, sin tener que pasar por los controles de Login de la aplicación, haciendo clic en crear usuario dentro del apartado usuarios.

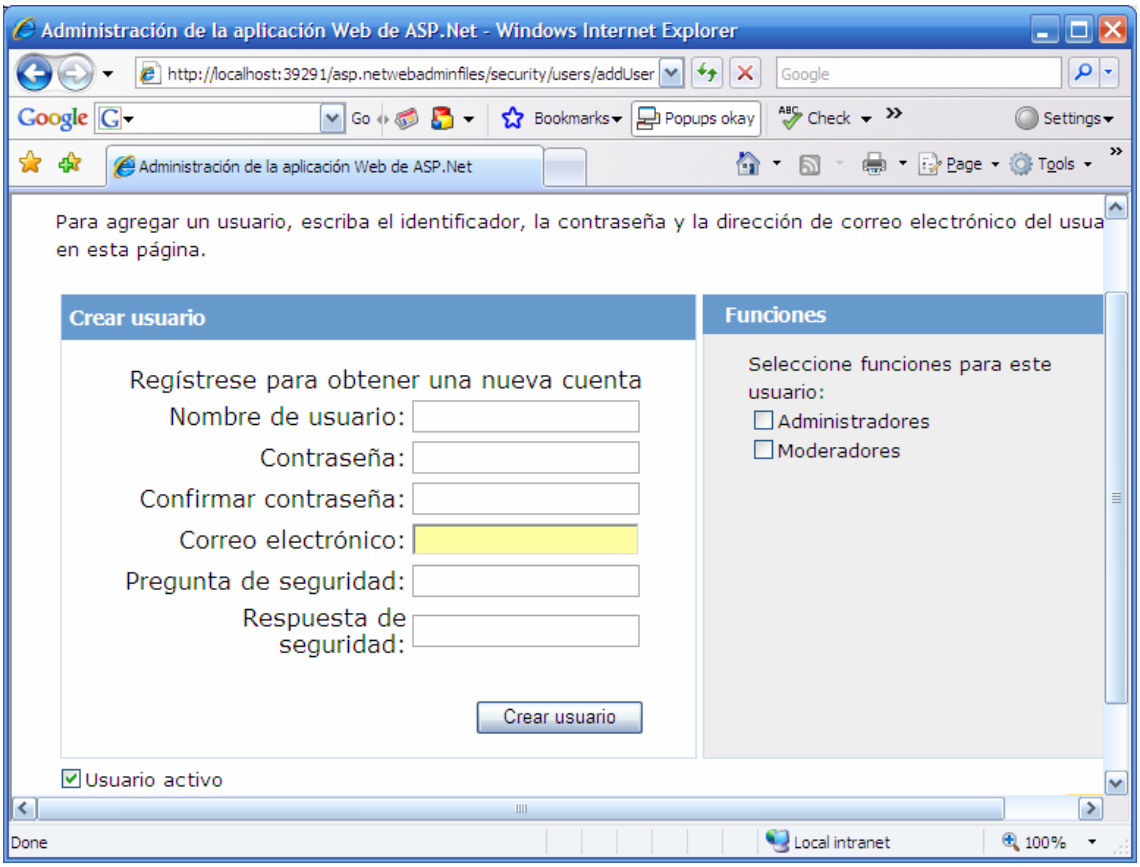


Fig. 5 Crear Usuario.

Hemos configurado la seguridad para nuestro sitio. Los cambios y modificaciones realizadas se han reflejado en el archivo web.config dentro de nuestro proyecto. Si refrescamos nuestro proyecto en el Solution Explorer podremos visualizar este archivo, así como un archivo ASPNETDB.mdb bajo la carpeta App_Data.

2. PÁGINAS MAESTRAS

2.1. Descripción

Una página principal o master page es una página especial que funciona como contenedor de plantillas y página de combinación de las páginas de contenido de una aplicación Web ASP.NET. Las páginas maestras constituyen un método muy útil para compartir estructura y contenido entre varias páginas de contenido. Se utilizan marcadores de posición de contenido para definir las secciones de la página principal que se deben reemplazar con contenido de las páginas de contenido.

Al utilizar una página maestra y las páginas de contenido relacionadas, se agregan las etiquetas de documento XHTML necesarias (por ejemplo html, head y body) únicamente en la página principal, y se dejan de crear los demás archivos .aspx (páginas ASP.NET) como páginas Web independientes. Las páginas de contenido definen el contenido que se debe insertar en los marcadores de posición de la página principal.

Cuando se realiza la solicitud HTTP de una página en tiempo de ejecución, la página principal y las de contenido se combinan en una sola clase con el mismo nombre que las páginas de contenido. La clase compilada y combinada resultante se deriva de la clase Page.

La principal ventaja es que, por ejemplo, si todas las páginas deben tener los mismos banners de cabecera y pie de página o el mismo menú de navegación, podemos definir esto una vez en una página maestra, de forma que todas las páginas asociadas a dicha página principal heredarán estos elementos comunes. De esta manera evitamos la duplicación innecesaria de código para estructuras o comportamientos del sitio que son compartidos, y a su vez mejoramos el mantenimiento de nuestro sitio Web.

La definición de una página maestra o principal es como la de cualquier página, con la única diferencia de que se guarda con una extensión .master. Las páginas maestras pueden incluir, controles, código, o cualquiera de los elementos con los que estamos acostumbrados a trabajar en una página. Sin embargo, una Master Page puede contener un tipo especial de control llamado ContentPlaceHolder. Un ContentPlaceHolder define una región de la representación de

la Master Page que puede sustituirse por el contenido de una página asociada a la maestra. Un ContentPaceHolder también puede contener contenido por defecto, por si la página derivada no necesita sobreescribir este contenido. Una página principal puede contener marcado directo y controles de servidor, además de controles contenedor.

Cada una de las páginas de contenido (Content Page) relacionada con la página principal debe hacer referencia a la página principal en un atributo MasterPageFile de su directiva @ Page. Las páginas de contenido sólo pueden tener esa directiva @ Page y uno o varios controles Content. Todo el texto, el marcado y los controles de la página se deben ubicar en controles Content, o de lo contrario se producirá un error.

El control ContentPlaceHolder de una página principal a la que está asociado un control Content se identifica estableciendo la propiedad ContentPlaceHolderID del control Content.

La directiva @ Page además de enlazar la página de contenido a una página principal concreta, define también un título para la página que se combinará en la página principal. Hay que tener en cuenta que la página de contenido no incluye otro marcado fuera de los controles Content. La página principal debe contener un elemento head con el atributo runat="server" para que se pueda combinar la configuración del título en tiempo de ejecución.

```
<%@ Page Language="C#" MasterPageFile="~/Página_Maestra.master"
AutoEventWireup="true" CodeFile="Default.aspx.cs" Inherits="_Default"
Title="Programación Orientada a la Web" %>
```

Podemos crear varias páginas principales para definir diseños distintos para partes diferentes del sitio, y un conjunto diferente de páginas de contenido para cada página principal. Así pues, todos los elementos ubicados en la página principal fuera de un control ContentPlaceHolder se procesan en todas las páginas que resultan de la combinación de la página principal y las páginas de contenido.

En tiempo de ejecución, el contenido de cada control Content de la página solicitada se combina con la página principal en la ubicación exacta del control ContentPlaceHolder relacionado de cada control. El resto del marcado y de los controles de la página principal no se

ven afectados. Los controladores de eventos pueden encontrarse en la clase principal y en la página de contenido.

La clase MasterPage está asociada a los archivos que tienen la extensión .master. Estos archivos se compilan en tiempo de ejecución como objetos MasterPage y se almacenan en la memoria caché del servidor.

El elemento de paginación especial se pone a disposición de la página de contenido a través de la propiedad Master de la clase base Page. La propiedad Master devuelve la instancia de la página principal; sin embargo, tiene como tipo la clase base MasterPage. Para tener acceso a los controles, propiedades y funciones de la página principal, la propiedad Master se puede convertir en MasterPage. El nombre de clase de la página principal se define mediante el atributo ClassName de la directiva @ Master.

La directiva @ Master puede contener prácticamente las mismas directivas que una directiva @ Control. Por ejemplo, la directiva de la página principal siguiente incluye el nombre de un archivo de código subyacente y asigna un nombre de clase a la página principal.

```
<%@ Master Language="C#" AutoEventWireup="true"
CodeFile="Página_Maestra.master.cs" Inherits="Página_Maestra" %>
```

2.2. ¿Cómo trabajar con páginas maestras?

Para agregar una página principal a un proyecto, pulsaremos con el botón derecho sobre el mismo y haremos clic en la opción Agregar nuevo elemento. Una vez que se nos abra la siguiente ventana, elegiremos el elemento Página principal (Ver **Fig. 6**). Por defecto, la página maestra creada contiene un control ContentPlaceHolder.

Si por el contrario lo que deseamos es crear una página de contenido derivada de una página maestra, debemos hacer clic con el botón derecho sobre nuestra página maestra, y seleccionar Agregar una página de contenido.

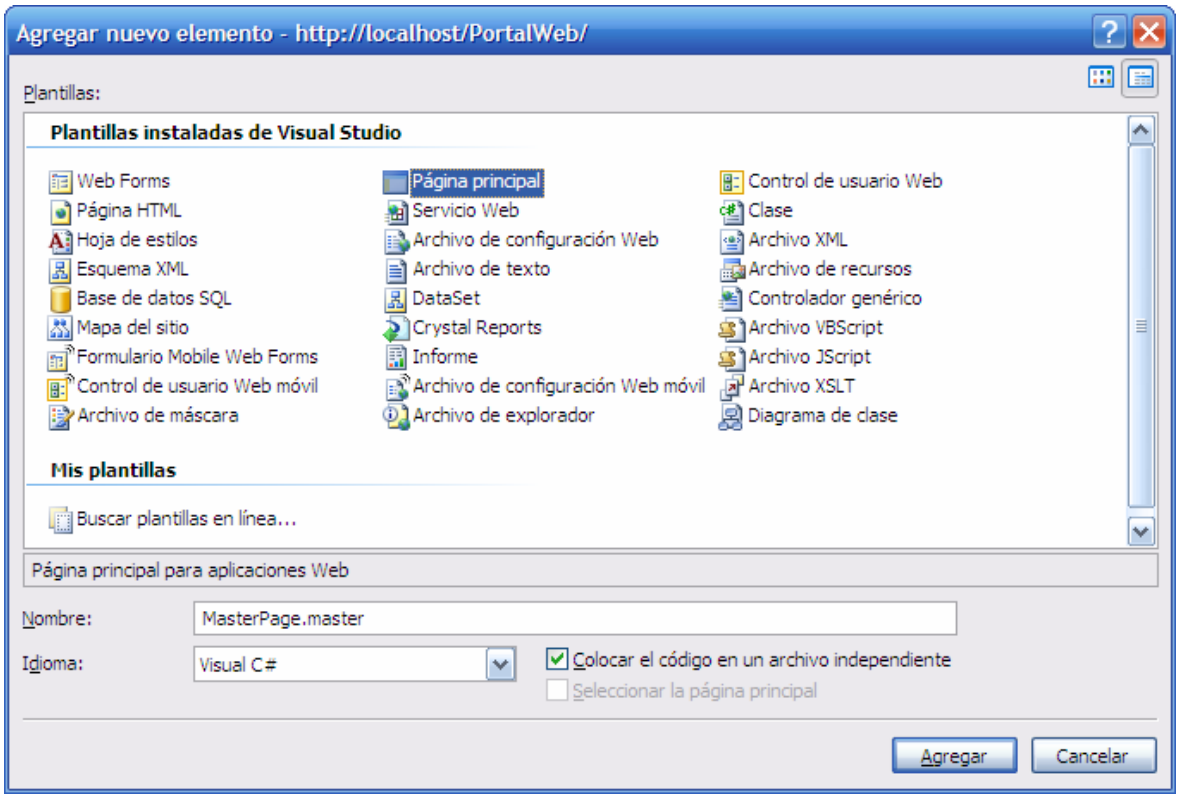


Fig. 6 Agregar una página maestra.

2.3. Ventajas de las páginas maestras

Las páginas principales proporcionan una funcionalidad que tradicionalmente los programadores creaban copiando el código, el texto y los elementos de control existentes repetidamente, mediante conjuntos de marcos, archivos de inclusión de elementos comunes, controles de usuario de ASP.NET, etc. Entre las ventajas de las páginas principales se incluyen las siguientes:

- ⇒ Permiten centralizar las funciones comunes de las páginas para que las actualizaciones puedan llevarse a cabo en un solo lugar.
- ⇒ Facilitan la creación de un conjunto de controles y código, y aplican los resultados en un conjunto de páginas. Por ejemplo, puede utilizar los controles en la página principal para crear un menú que se aplique a todas las páginas.
- ⇒ Proporcionan un control más preciso sobre el diseño de la página final al permitir controlar el modo en que se representan los controles PlaceHolder.

- ⇒ Proporcionan un modelo de objetos que permite personalizar la página principal a partir de páginas de contenido individuales.
- ⇒ Evitan la duplicidad innecesaria de gran parte de código.

2.4. Comportamiento en tiempo de ejecución

En tiempo de ejecución, las páginas principales se controlan en la secuencia siguiente:

- ⇒ Los usuarios solicitan una página escribiendo la dirección URL de la página de contenido.
- ⇒ Cuando se obtiene la página, se lee la directiva @ Page. Si la directiva hace referencia a una página principal, también se lee la página principal. Si las páginas se solicitan por primera vez, se compilan las dos páginas.
- ⇒ La página principal con el contenido actualizado se combina en el árbol de control de la página de contenido.
- ⇒ El contenido de los controles Content individuales se combina en el control ContentPlaceHolder correspondiente de la página principal.
- ⇒ La página combinada resultante se representa en el explorador.

Desde la perspectiva del usuario, la combinación de las páginas principales y de contenido da como resultado una única página. La dirección URL de esta página es la de la página de contenido. Desde la perspectiva del programador, las dos páginas actúan como contenedores diferentes para sus respectivos controles. La página de contenido actúa como un contenedor de la página principal. Sin embargo, se puede hacer referencia a los miembros públicos de una página principal a partir del código de la página de contenido. Hemos de tener en cuenta que la página principal pasa a formar parte de la página de contenido. De hecho, la página principal actúa fundamentalmente de igual forma que un control de usuario: como un elemento secundario de la página de contenido y como un contenedor dentro de esa página.

3. TEMAS Y MÁSCARAS DE ASP .NET 2.0

3.1. Introducción

Los temas son una colección de propiedades que permiten definir el aspecto de las páginas y de los controles de un sitio Web. Concretamente, un tema o Theme es un grupo de ficheros que puede estar compuesto por archivos de máscara, archivos de hojas de estilos en cascada (CSS), imágenes y otros recursos. Aplicando un tema podemos dar un aspecto coherente a las páginas de una aplicación Web, en toda una aplicación Web o en todas las aplicaciones Web de un servidor. Además obtenemos la ventaja de poder administrar los cambios de estilo mediante la realización de cambios en el tema, sin tener que editar las páginas de forma individual.

3.2. Máscaras de temas y controles

Máscaras

Los archivos de máscara o skins son archivo con extensión .skin que contienen los valores de las propiedades de los controles individuales de servidor Web de ASP.NET, como pueden ser Button, Label, TextBox o Calendar.

La configuración de las máscaras de control es parecida al propio marcado del control, pero únicamente contiene las propiedades que se desee establecer como parte del tema.

Estos archivos .skin se crean en la carpeta Theme. Un archivo .skin puede contener una o más máscaras de control para uno o más tipos de control. Existe la posibilidad de definir un archivo de máscaras independiente para cada control o definir todas las máscaras para un tema en un archivo único.

Existen dos tipos de máscaras de control: máscaras predeterminadas y máscaras con nombre:

⇒ Las **máscaras predeterminadas** se aplican automáticamente a todos los controles del mismo tipo cuando un tema se aplica a una página. Una máscara de control es predeterminada si no tiene un atributo SkinID. Por ejemplo, si se crea una máscara

predeterminada para un control Calendar, la máscara de control se aplicará a todos los controles Calendar de las páginas en las que se utilice el tema. Este tipo de máscaras coincide exactamente atendiendo al tipo de control.

⇒ Las **máscaras con nombre** son máscaras de controles con un conjunto de propiedades SkinID. Las máscaras con nombre no se aplican automáticamente a todos los controles según el tipo. En su lugar, una máscara con nombre se aplica explícitamente a un control estableciendo la propiedad SkinID del control. Al crear máscaras con nombre, se pueden configurar diferentes máscaras para distintas instancias del mismo control en una aplicación.

Hojas de estilo

Un tema también puede incluir una hoja de estilos en cascada (archivo .css). Cuando colocamos un archivo .css en la carpeta de temas, la hoja de estilos se aplica automáticamente como parte del tema. La hoja de estilos se define utilizando la extensión de nombre de archivo .css en la carpeta de tema.

Gráficos del tema y otros recursos

Los temas también pueden incluir gráficos y otros recursos, como archivos de secuencias de comandos o archivos de sonido. Normalmente, estos archivos están en la misma carpeta que los archivos de máscara de dicho tema, pero pueden estar en cualquier parte de la aplicación Web, por ejemplo, en una subcarpeta de la carpeta de temas.

3.3. Ámbito de los temas

Podemos definir los temas para una aplicación Web única o como temas globales que pueden utilizar todas las aplicaciones en un servidor Web. Una vez definido un tema, se puede colocar en páginas individuales utilizando el atributo Theme o StyleSheetTheme de la directiva @Page, o se puede aplicar a todas las páginas de una aplicación configurando el elemento <pages> en el esquema de configuración de ASP.NET, es decir, en el archivo de configuración de la aplicación (web.config). Si dicho elemento de define en el archivo Machine.config, el tema se aplicará a todas las páginas de las aplicaciones Web en el servidor.

Temas de páginas

Un tema de página corresponde a una carpeta de temas con máscaras de control, hojas de estilos, archivos de gráficos y otros recursos creados como una subcarpeta de la carpeta \App_Themes en un sitio Web. Cada tema constituye una subcarpeta diferente con respecto a la carpeta \App_Themes. En la siguiente figura podemos ver el contenido de una carpeta \App_Themes con cuatro temas.

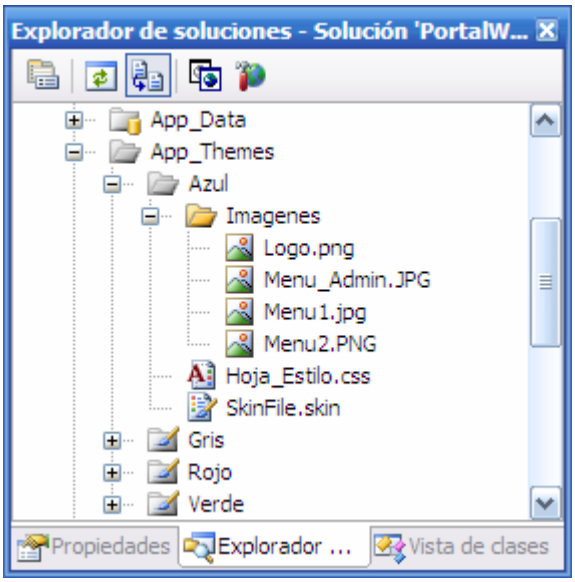


Fig. 7 Temas del sitio Web.

Temas globales

Un tema global es un tema que puede aplicar a todos los sitios Web en un servidor. Los temas globales permiten definir una apariencia de conjunto para un dominio cuando se mantienen varios sitios Web en el mismo servidor.

Los temas globales se parecen a los de página en que ambos tipos incluyen valores de propiedades, la configuración de las hojas de estilos y gráficos. Sin embargo, los temas globales se almacenan en una carpeta denominada Themes que es global al servidor Web. Cualquier sitio Web del servidor y cualquier página de cualquier sitio Web pueden hacer referencia a un tema global.

3.4. Prioridad en la configuración de temas

Se puede especificar la prioridad que tiene la configuración del tema sobre la configuración regional del control especificando cómo se aplica el tema.

Si establece la propiedad Theme de una página, los valores de control en el tema y la página se combinan para formar la configuración final para el control. Si la configuración del control se

define tanto en el control como en el tema, la configuración del control del tema reemplaza cualquier configuración de la página en el control. Esta estrategia hace posible que el tema pueda crear una apariencia coherente a lo largo de las páginas, incluso si los controles de las páginas ya tuvieran valores de propiedades individuales.

Del mismo modo, es posible aplicar un tema como tema de la hoja de estilos estableciendo la propiedad StyleSheetTheme de la página. En este caso, la configuración de la página local tiene prioridad sobre aquellos definidos en el tema cuando la configuración se define en ambos lugares. Éste es el modelo utilizado en las hojas de estilos en cascada. Se podría aplicar un tema como tema de la hoja de estilos si se desea poder establecer las propiedades de controles individuales en la página mientras se sigue aplicando un tema para lograr una apariencia de conjunto.

Los elementos de temas globales no pueden reemplazarse parcialmente por elementos de temas de nivel de aplicación. Si creamos un tema de nivel de aplicación con el mismo nombre que un tema global, los elementos de tema de nivel de aplicación no reemplazarán los elementos del tema global.

3.5. Propiedades definibles mediante temas

Como regla general, se pueden usar los temas para definir las propiedades relacionadas con la apariencia de una página o de un control o el contenido estático. Sólo se pueden establecer esas propiedades que tienen un atributo ThemeableAttribute establecidas como true en la clase de control.

Las propiedades que especifican explícitamente el comportamiento de los controles en lugar de su apariencia no aceptan valores de tema. Por ejemplo, no podemos configurar la propiedad CommandName de un control Button mediante un tema. Igualmente, no podemos utilizar un tema para configurar la propiedad AllowPaging o DataSource de un control GridView.

Por otro lado no podemos utilizar generadores de expresiones, que generan expresiones de código para asignarlas a una página en tiempo de compilación, en temas o máscaras.

3.6. Temas frente a CSS

Tanto los temas como las hojas de estilos definen una serie de atributos comunes que se pueden aplicar a cualquier página. Sin embargo, los temas se diferencian de las hojas de estilos en los siguientes puntos:

- ⇒ Los temas pueden definir muchas propiedades de un control o de una página, y no sólo las propiedades de un estilo. Por ejemplo, los temas permiten especificar los gráficos de un control TreeView, el diseño de plantilla de un control GridView, etcétera.
- ⇒ Los temas pueden incluir gráficos.
- ⇒ Los temas no se muestran en cascada del modo en que lo hacen las hojas de estilos. De forma predeterminada, cualquier valor de propiedad definido en un tema al que se haga referencia en la propiedad Theme de una página reemplazará los valores de las propiedades definidos mediante declaración, a menos que aplique explícitamente el tema mediante la propiedad StyleSheetTheme.
- ⇒ Sólo se puede aplicar un tema a cada página. No puede aplicar varios temas a una página, a diferencia de las hojas de estilos que sí se pueden aplicar varias.

3.7. Consideraciones de seguridad

Los temas pueden causar problemas de seguridad cuando se utilizan en el sitio Web. Debemos tener cuidado ya que se pueden utilizar temas malintencionados para:

- ⇒ Modificar el comportamiento de un control de forma que no se comporte según lo previsto.
- ⇒ Insertar secuencias de comandos en el cliente, lo que puede suponer un riesgo de creación de secuencias de comandos entre sitios.
- ⇒ Modificar la validación.
- ⇒ Divulgar información confidencial.

Las formas de mitigar estas amenazas comunes son las siguientes:

- ⇒ Proteger los directorios de temas globales y de aplicación con una configuración de control de acceso apropiada. Sólo los usuarios de confianza deben poder escribir archivos en los directorios de temas.
- ⇒ No utilizar temas de un origen que no sea de confianza.

⇒ No exponer el nombre del tema en los datos de una consulta. Los usuarios malintencionados podrían utilizar esta información para usar temas que el programador no conoce y, de ese modo, divulgar información confidencial.

3.8. Cómo definir temas en ASP .NET

3.8.1. Para crear un tema de página

a. Creamos una nueva carpeta denominada App_Themes en nuestro sitio Web.
b. A continuación, creamos una nueva subcarpeta de la carpeta App_Themes para que contenga los archivos de tema. El nombre de la subcarpeta coincidirá con el nombre del tema. Por ejemplo, para crear un tema denominado Azul, crearemos una carpeta denominada \App_Themes\Azul (Ver **Fig. 7**).
c. Agregamos archivos a la carpeta para incorporar las máscaras, hojas de estilos e imágenes que formarán el tema.

3.8.2. Para crear una máscara

a. Tenemos dos maneras de hacerlo, podemos crear un archivo de texto nuevo en la subcarpeta de tema con la extensión .skin. O con el botón derecho sobre la subcarpeta de tema elegir la opción agregar nuevo elemento y a continuación seleccionar la plantilla Archivo de máscara.
b. En el archivo .skin, agregamos una definición normal de control (con sintaxis declarativa), pero incluyendo únicamente las propiedades que vaya a establecer para el tema y sin incluir el atributo ID. La definición del control debe incluir el atributo runat="server".

En el ejemplo siguiente se muestra una máscara de control predeterminada para un control Label en la que se define el color de todos los controles Label del tema:

```
<asp:Label runat="server" SkinId = "titulosmaster" ForeColor = "Navy"/>
```

Una forma cómoda de crear una máscara consiste en agregar el control a una página y configurarlo hasta conseguir el aspecto deseado. Por ejemplo, podríamos agregar un control Calendar a una página y establecer su encabezado de días, de fecha seleccionada y otras

propiedades. A continuación, podemos copiar la definición del control de la página a un archivo de máscara y quitar luego el atributo ID. Para cada máscara de controles que queramos crear repetiríamos los pasos a y b.

3.9. Cómo aplicar temas en ASP.NET

Como hemos visto podemos aplicar los temas a una página, un sitio Web, o globalmente. Al establecer un tema en el nivel del sito Web los estilos y máscaras se aplican a todas las páginas y controles del sitio, a no ser que reemplace un tema para una página individual. Al establecer un tema en el nivel de la página, los estilos y máscaras se aplican a esa página y a todos sus controles. De manera predeterminada, los temas reemplazan la configuración local del control. Como alternativa, también se puede establecer un tema como tema de la hoja de estilos, de forma que dicho tema se aplique sólo a la configuración del control que no esté específicamente establecida en el mismo.

3.9.1. Para aplicar un tema a un sitio Web

En el archivo Web.config de la aplicación, establecemos el elemento <pages> en el nombre del tema, ya sea éste global o de página, como se muestra en el siguiente ejemplo:

```
<configuration>
   <system.web>
       <pages theme ="Azul">
     </pages>
   </system.web>
</configuration>
```

Cuando un tema de aplicación tiene el mismo nombre que otro global, la prioridad recae en el tema de página. Para establecer un tema como tema de la hoja de estilos de forma que se subordine a la configuración local del control, establecemos en su lugar el atributo StyleSheetTheme:

```
<configuration>
    <system.web>
          <pages StyleSheetTheme ="Azul">
     </system.web>
</configuration>
```

Los temas que se establecen en el archivo Web.config se aplican a todas las páginas Web ASP.NET de esa aplicación. La configuración del tema en el archivo Web.config sigue las convenciones normales de jerarquía. Por ejemplo, para aplicar un tema únicamente a un subconjunto de páginas, podemos colocar estas páginas en una carpeta con su propio archivo Web.config o bien crear un elemento <location> en el archivo Web.config raíz para especificar una carpeta.

3.9.2. Para aplicar un tema a una página

Establecemos el atributo Theme o StyleSheetTheme de la directiva @Page en el nombre del tema que se va a utilizar, como se muestra en el siguiente ejemplo:

```
<%@ Page Theme="Azul" %>
<%@ Page StylesheetTheme="Azul" %>
```

3.9.3. Cómo aplicar máscaras a los controles

Las máscaras definidas en el tema se aplicarán en todas las instancias de controles de la aplicación o en las páginas a las que se aplique el tema. En algunos casos, es posible que deseemos aplicar un conjunto concreto de propiedades a un control individual. Puede hacerse creando una máscara con nombre (entrada en un archivo .skin que tiene establecida la propiedad SkinID) y aplicándola a continuación mediante el identificador a controles individuales.

Para aplicar una máscara con nombre a un control, establecemos la propiedad SkinID del control, como se muestra en el ejemplo siguiente:

```
<asp:Label ID="Label1" runat="server" SkinID="titulosmaster">
</asp:Label>
```

Si el tema de la página no incluyera ninguna máscara de controles coincidente con la propiedad SkinID, el control utilizará la máscara predeterminada para ese tipo de control.

4. PERSONALIZACIÓN DE SITIOS WEB: PERFILES

4.1. Introducción

En muchas aplicaciones es necesario almacenar y utilizar información que es única de un usuario. De tal forma que cuando este visita el sitio, se puede utilizar dicha información para presentar al usuario una versión personalizada de la aplicación Web.

La personalización de una aplicación requiere varios elementos: la información se debe almacenar con un identificador de usuario único, ha de ser posible reconocer a los usuarios cada vez que visiten el sitio y, a continuación, buscar la información del usuario que se necesite. La característica de perfiles de ASP.NET realiza automáticamente todas estas tareas.

La característica de perfiles de ASP.NET asocia información con cada usuario y la almacena en un formato persistente. Los perfiles nos permiten administrar la información del usuario sin que sea necesario crear y mantener una base de datos especifica para ello. Además, esta característica permite que la información del usuario esté disponible a través de una API con establecimiento inflexible de tipos a la que se puede tener acceso desde cualquier parte de la aplicación.

La característica de perfiles proporciona una función de almacenamiento genérico que permite definir y mantener casi cualquier tipo de datos, al tiempo que los datos están disponibles con seguridad de tipos.

4.2. Funcionamiento de los perfiles en ASP.NET

Para poder utilizar perfiles, en primer lugar debemos habilitarlos modificando el archivo de configuración de la aplicación Web ASP.NET. Como parte de la configuración, deberemos especificar un proveedor de perfiles, que es la clase subyacente que realiza las tareas de nivel inferior consistentes en almacenar y recuperar los datos del perfil. En este caso utilizaremos el proveedor de perfiles incluido con .NET Framework, que almacena los datos de los perfiles en SQL Server, aunque también tendríamos la posibilidad de crear y utilizar nuestro propio proveedor de perfiles. Podemos especificar una instancia de la clase SqlProfileProvider que se

conecte a una base de datos de nuestra elección, o bien utilizar la instancia predeterminada de SqlProfileProvider que almacena los datos del perfil en el servidor Web local.

La característica de perfiles se configura definiendo la lista de propiedades cuyos valores desea mantener. Por ejemplo, en nuestra aplicación vamos a querer almacenar el tema preferido de cada usuario para posteriormente personalizar la aplicación en función de esta elección.

De tal forma que en el archivo de configuración definiremos una propiedad de perfil denominada Tema. Con lo que, la sección profile del archivo de configuración tendrá la siguiente apariencia:

```
<profile>
        <properties>
            <add name="Tema"/>
        </properties>
</profile>
```

Cuando se ejecuta la aplicación, ASP.NET crea una clase ProfileCommon, que es una clase generada dinámicamente que hereda la clase ProfileBase. La clase ProfileCommon dinámica incluye las propiedades creadas a partir de las definiciones de propiedades de perfil especificadas en la configuración de la aplicación. A continuación, una instancia de esta clase dinámica ProfileCommon se establece como valor de la propiedad Profile del objeto HttpContext actual y está disponible para las páginas de la aplicación.

En la aplicación, recopilaremos el valor que deseamos almacenar, en este caso el tema preferido del usuario y se lo asignaremos a la propiedad de perfil que hemos definido. En nuestro caso, en la plantilla para usuarios autenticados del control LoginView tendremos un control DropDownList para que el usuario elija el tema que se aplicará al sitio Web cuando éste inicie sesión *(Ver Capítulo 3, Sección Personalización del Sitio Web)*.

No es necesario determinar de forma explícita la identidad del usuario ni realizar búsquedas en ninguna base de datos. Únicamente hay que obtener el valor de la propiedad de un perfil a fin de que ASP.NET realice las acciones necesarias para identificar al usuario actual y busque el valor en el almacén de perfiles persistente.

4.3. Identificación de usuario para las propiedades de perfil en ASP .NET

La característica de perfiles de usuario de ASP.NET se ha diseñado para proporcionar información que es únicamente del usuario actual. Los perfiles pueden funcionar con usuarios autenticados o con usuarios anónimos (sin autenticar). En nuestra aplicación solo nos interesará el funcionamiento con usuarios autenticados.

Información sobre la definición de propiedades

Cuando se define una propiedad en el perfil, se especifica el nombre que se utilizará para hacer referencia a la misma. Por ejemplo, en nuestro caso asignaremos el nombre Tema a la propiedad de perfil y, a continuación, podremos obtener y establecer el valor de la misma como Profile.Tema.

Opcionalmente se pueden definir las características adicionales siguientes para cada propiedad:

- ⇒ **Type:** Especifica el tipo de la propiedad. El tipo predeterminado es String. Puede especificar cualquier clase de .NET como tipo (Int32, DateTime, StringCollection, etc.). Si el tipo no está definido en .NET Framework, debemos asegurarnos de que la aplicación Web tenga acceso a él. Podemos incluir el ensamblado compilado del tipo en el directorio Bin del sitio Web o en la Caché de ensamblados global (GAC) o podemos colocar el código fuente del tipo en el directorio App_Code del sitio Web.

- ⇒ **serializeAs:** Especifica el formateador de serialización (serialización de cadena, binaria, XML o específica del proveedor). La serialización predeterminada es la de cadena.

- ⇒ **allowAnonymous:** Especifica un valor booleano que indica si la propiedad se administra para los usuarios anónimos. De forma predeterminada, el valor es false. Si deseásemos que la propiedad esté disponible para los usuarios no autenticados, estableceríamos la propiedad en true.

- ⇒ **defaultValue** Especifica un valor con el que se inicializa la propiedad.

- ⇒ **readOnly** Especifica un valor booleano que indica si se puede modificar la propiedad.

⇒ **provider** Especifica un proveedor específico para la propiedad. De manera predeterminada, todas las propiedades se administran utilizando el proveedor predeterminado especificado para las propiedades de perfil, pero cada propiedad también puede utilizar otros proveedores.

⇒ **customProviderData** Especifica una cadena opcional que contiene información personalizada que se pasa al proveedor de perfiles. Cada proveedor puede implementar lógica personalizada para el uso de estos datos.

Además, las propiedades de perfil pueden organizarse como grupos de propiedades mediante el elemento de configuración group.

Teniendo en cuenta estas propiedades definiremos nuestra propiedad de perfil de la siguiente manera:

```
<profile>
        <properties>
                <add name="Tema" defaultValue="Azul" type="String"/>
        </properties>
</profile>
```

De esta forma, la primera vez que un usuario inicie sesión y las siguientes si no cambia su propiedad de perfil, el tema por defecto será “Azul”.

5. ADMINISTRACIÓN DE MIEMBROS Y FUNCIONES

5.1. Introducción

La versión de Microsoft Visual Studio 2005 no trae consigo una solución rápida, tal como un control Web, para el mantenimiento de las bases de datos de usuarios (pertenencia) y funciones o roles de Microsoft IIS. Esto es un problema a la hora de mover nuestra aplicación del entorno de desarrollo al servidor IIS. La utilidad de configuración Web de ASP.Net que nos ofrece Microsoft sólo se puede ejecutar en un entorno de desarrollo (desde el propio MS Visual Studio).

Para poder mantener y configurar nuestra base de datos referente a los usuarios y roles vamos a hacer uso del diseño realizado por el prestigioso desarrollador Peter Kellner (http://peterkellner.net). Dicha solución se adapta a nuestra estructura de tres niveles o capas que permite administrar miembros y funciones con herramientas estándar de Microsoft ASP.NET. Se trata de una solución flexible que se puede agregar fácilmente a cualquier proyecto de sitio Web de ASP.NET 2.0 existente. Lo primero que haremos será descargarnos el código de la solución propuesta por Peter Kellner. Para ello acudiremos a este enlace: http://download.microsoft.com/download/1/6/0/160CE638-446F-428B-BF22-8D7C5A05FB47/MembershipEditor.msi

A continuación procederemos a su instalación. Cuando hayamos terminado, en la carpeta destino tendremos lo siguiente:

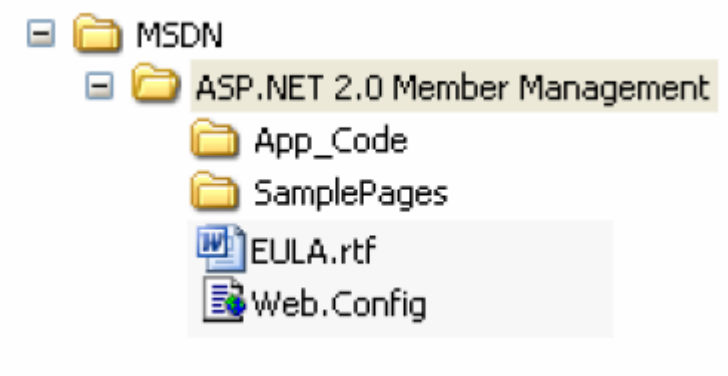


Fig. 8 MemberShip.

Carpeta App_Code: donde se encuentra los archivos MembershipUserODS.cs, MembershipUserWrapper.cs y RoleDataObject.cs. Estos serán los ficheros que copiaremos en nuestro proyecto, en la carpeta llamada de lógica de negocio (Business Logic) dentro de App_Code.

Carpeta SamplePages: contiene varios ejemplos que muestran el uso de las clases albergadas en App_Code. Estos ejemplos son los archivos DetailsViewOneUser.aspx, DetailsViewOneUser.cs, DetailsViewSample.aspx, DetailsViewSample.cs, GridViewSample.aspx, GridViewSample.cs, Membership.aspx y Membership.cs.

El fichero EULA.rtf. El fichero de las licencias.

El fichero Web.Config. *Es el fichero de configuración del proyecto que acabamos de instalar.*

Lo siguiente que vamos a hacer es tomar los ficheros de la carpeta App_Code y copiarlos en nuestro proyecto, en la carpeta App_Code/Business Logic. La página Membership.aspx descargada la podríamos utilizar directamente para seleccionar, actualizar, insertar y eliminar miembros y funciones, así como para asignar funciones a miembros. Pero haremos nuestra propia versión, muy similar, pero no idéntica.

Para que estos archivos funcionen de manera adecuada hemos de tener la aplicación configurada para el uso de servicio de suscripciones. Si no fuera así y fuera la primera implementación de administración de pertenencia y funciones en la aplicación, procederíamos de la siguiente manera:

- ⇒ En el menú, haríamos clic en Sitio Web / Configuración de ASP.NET.

- ⇒ Seguiríamos los pasos del asistente (1 a 7) para crear varios usuarios y funciones de ejemplo. Se creará un archivo web.config válido en el proyecto actual con la información suficiente como para que la administración de miembros esté preparada y en funcionamiento. De forma predeterminada, utilizará la configuración de SQL Server Express 2005.

⇒ Incluiremos los tres archivos .cs en el proyecto y, a continuación, las páginas .aspx de ejemplo para verificar un buen funcionamiento del mismo.

A continuación describiremos brevemente el esquema de funcionamiento. En esta ocasión el modelo de tres capas vendrá dado del modo siguiente. El primer nivel, es decir la capa de presentación, estará constituida por la página de ASP.NET (.aspx), que interactuará con dos objetos empresariales a través del origen de datos de objetos. Los objetos empresariales, por su parte, constituyen el nivel medio y contienen los miembros y las funciones. El tercer nivel, o servidor, constará de las API de administración de pertenencia y funciones que proporciona ASP.NET. Los objetos del nivel medio podrán ser colocados fácilmente en cualquier proyecto de ASP.NET 2.0 y utilizarse directamente, sin apenas cambios.

Es necesario tener en cuenta que la API de pertenencia que ofrece Microsoft utiliza su propia tecnología de proveedor. Es por ello que la solución que se presenta es independiente de la base de datos. Las entradas adecuadas para seleccionar dicho proveedor vendrán definidas en el archivo web.config.

5.2. Descripción

El proyecto necesario para ejecutar esta utilidad es muy simple e independiente. Sus archivos, disponibles para descarga, contienen un ejemplo operativo completo. Debido a que los usuarios y las funciones no disponen de acceso directo a la base de datos, todo lo que se debe hacer es tomar los tres objetos de datos incluidos: MembershipUserODS, MembershipUserWrapper, RoleDataObject.

En la carpeta SamplePages existen varios ejemplos más que muestran el uso de los módulos mencionados anteriormente. Por ejemplo, Membership.aspx se puede utilizar para seleccionar, actualizar, insertar y eliminar miembros y funciones, así como para asignar funciones a miembros.

En el caso de las aplicaciones de ASP.NET 2.0 que ya disponen de un módulo de pertenencia en funcionamiento, la configuración ya realizada en estas páginas debería ser suficiente.

A continuación vamos a describir cada una de las tres clases de las que se compone la solución propuesta.

Clase MembershipUserWrapper

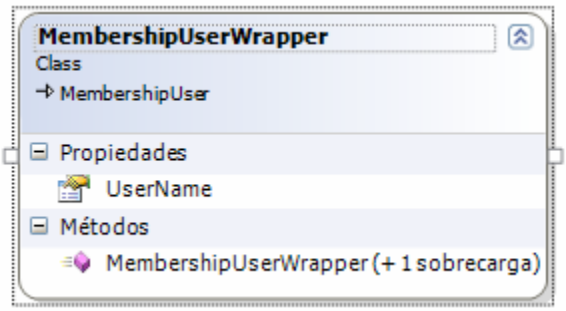


Fig. 9 Clase MembershipUserWrapper.

Esta clase hereda de la clase System.Web.Security.MembershipUser, con lo que la mayoría de las propiedades de esta clase son las mismas que las asociadas a MembershipUser. Entre estas propiedades se incluyen:

- ⇒ ProviderUserKey
- ⇒ UserName
- ⇒ LastLockoutDate
- ⇒ CreationDate
- ⇒ PasswordQuestion
- ⇒ LastActivityDate
- ⇒ ProviderName
- ⇒ IsLockedOut
- ⇒ Email
- ⇒ LastLoginDate
- ⇒ IsOnline
- ⇒ LastPasswordChangedDate
- ⇒ Comment

Es la clase encargada de albergar las propiedades correspondientes a un usuario.

Clase MembershipODS.cs

La clase de la capa de lógica de negocio que utilizaremos para manipular la pertenencia será MembershipODS.cs. Esta clase está diseñada para, mediante un objeto ObjectDataSource, tratar la pertenencia en nuestra aplicación. Veamos en detalle esta clase.

Fig. 10 Clase MembershipODS.

```
namespace MembershipUtilities
{
    /// <summary>
    /// Class Used to Select, Update, Insert and Delete
    /// From the Microsoft ASP.NET Membership API
    /// </summary>
    ///
    [DataObject(true)]  // This attribute allows the ObjectDataSource wizard to see this class
    public class MembershipUserODS
    {
        ..............
        ..............
```

Observamos que la declaración de esta clase viene precedida del atributo [DataObject(true)]. Con este atributo indicamos al asistente para la creación de ObjectDataSource de Visual Studio 2005 que a la hora de buscar objetos de datos en la clase de datos, busque sólo los miembros que contengan dicho atributo.

Esta clase es la utilizada para realizar las funciones correspondientes a Select, Update, Insert y Delete del API de Microsoft ASP.NET Membership. Sin entrar en detalle, vamos a ver cada uno de estos métodos.

Método Insert

```
[DataObjectMethod(DataObjectMethodType.Insert, true)]
        static public void Insert(string userName, bool isApproved,
            string comment, DateTime lastLockoutDate, DateTime creationDate,
            string email, DateTime lastActivityDate, string providerName, bool isLockedOut,
            DateTime lastLoginDate, bool isOnline, string passwordQuestion,
            DateTime lastPasswordChangedDate, string password, string passwordAnswer)
        {
            MembershipCreateStatus status;
            Membership.CreateUser(userName, password, email, passwordQuestion,
passwordAnswer, isApproved, out status);
            if (status != MembershipCreateStatus.Success)
            {
                throw new ApplicationException(status.ToString());
            }
            MembershipUser mu = Membership.GetUser(userName);
            mu.Comment = comment;
            Membership.UpdateUser(mu);
        }
```

Como vemos este método hace uso de la clase MemberShip del espacio de nombre System.Web.Security. Este método es el empleado para insertar un nuevo miembro en la base de datos.

Método Delete

Es el más sencillo de todos y toma un solo parámetro, UserName.

```
[DataObjectMethod(DataObjectMethodType.Delete, true)]
        static public void Delete(string UserName)
        {
            Membership.DeleteUser(UserName, true);
        }
```

Método Select

Para realizar la operación SELECT, la clase cuenta con un método (GetMembers()) que admite múltiples sobrecargas.

Todos los métodos Select devuelven una colección genérica de tipos. Este tipo de devolución es de la clase MemberShipUserWrapper.cs, es decir, devuelven List<MembershipUserWrapper.cs>. Los tipos genéricos se utilizan porque el origen ObjectDataSource asociado en última instancia a la clase utiliza la reflexión para determinar los nombres y tipos de columna. Estos nombres y tipos están asociados a cada una de las filas de

datos devueltas del mismo modo que SqlDataSource utiliza los metadatos de base de datos de una tabla o procedimiento almacenado para determinar los nombres de las columnas de cada una de las filas.

```
/// <summary> ...
[DataObjectMethod(DataObjectMethodType.Update, false)]
static public void Update(string Username, bool isApproved)...

/// <summary> ...
[DataObjectMethod(DataObjectMethodType.Select, false)]
static public List<MembershipUserWrapper> GetMembers()...

/// <summary> ...
[DataObjectMethod(DataObjectMethodType.Select, true)]
static public List<MembershipUserWrapper> GetMembers(string sortData)...

/// <summary> ...
[DataObjectMethod(DataObjectMethodType.Select, false)]
static public List<MembershipUserWrapper> GetMembers(bool approvalStatus, string sortData)...

/// <summary> ...
[DataObjectMethod(DataObjectMethodType.Select, false)]
static public List<MembershipUserWrapper> GetMembers(bool returnAllApprovedUsers,
    bool returnAllNotApprovedUsers,
    string usernameToFind, string sortData)...
```

En realidad, los tres primeros métodos GetMembers() llaman al último, añadiendo unos valores por defecto a determinados parámetros.

Criterios de ordenación personalizados

En alguna de las sobrecargas del método GetMembers() podemos pasar una cadena de parámetros llamada sortData. Si en la declaración de ObjectDataSource se especifica SortParameterName como uno de los atributos, este parámetro se pasará automáticamente a todos los métodos Select. Su valor será el nombre especificado por el atributo SortExpression en la columna del control de datos.

El método Comparar se invoca en función del parámetro sortName del método GetMembers. Dado que estas páginas Web de ASP.NET no tienen estado, es preciso asumir que la dirección del orden actual (hacia adelante o hacia atrás) se almacena en el estado de vista. Cada llamada invierte la dirección de la anterior. Es decir, la dirección cambia entre orden hacia adelante y orden hacia atrás cuando el usuario hace clic en el encabezado de la columna.

Teniendo en cuenta que el diseño de esta clase está pensado para su uso en un control GridView, el parámetro que se pasa a GetMembers(sortData) contiene los datos del atributo SortExpression de la columna de GridView. Al realizar una solicitud de ordenación hacia atrás, la palabra "DESC" se anexa al final de la cadena de ordenación. De este modo, por ejemplo, la primera vez que el usuario hace clic en la columna Email, el elemento sortData pasado a GetMembers será "Email". La segunda vez, en cambio, el parámetro sortData pasará a ser "Email DESC", luego "Email", luego "Email DESC", y así sucesivamente. Cabe mencionar también que la primera vez que se carga la página, el parámetro sortData se pasa como cadena de longitud cero (no nula). A continuación se muestra el mecanismo interno del método GetMembers que recupera y ordena los datos para que se devuelvan en el orden correcto.

```
[DataObjectMethod(DataObjectMethodType.Select, false)]
      static public List<MembershipUserWrapper> GetMembers(bool
returnAllApprovedUsers, bool returnAllNotApprovedUsers,
         string usernameToFind, string sortData)
      {
         List<MembershipUserWrapper> memberList = new
List<MembershipUserWrapper>();
         if (usernameToFind != null)
         {
            MembershipUser mu = Membership.GetUser(usernameToFind);
            if (mu != null)
            {
               MembershipUserWrapper md = new MembershipUserWrapper(mu);
               memberList.Add(md);
            }
         }
         else
         {
            MembershipUserCollection muc = Membership.GetAllUsers();
            foreach (MembershipUser mu in muc)
            {
               if ((returnAllApprovedUsers == true && mu.IsApproved == true) ||
                   (returnAllNotApprovedUsers == true && mu.IsApproved == false))
               {
                  MembershipUserWrapper md = new MembershipUserWrapper(mu);
                  memberList.Add(md);
               }
            }
            if (sortData == null)
            {
               sortData = "UserName";
            }
            if (sortData.Length == 0)
            {
               sortData = "UserName";
            }
           string sortDataBase = sortData;
            string descString = " DESC";
```

```
if (sortData.EndsWith(descString))
{
    sortDataBase = sortData.Substring(0, sortData.Length - descString.Length);
}
Comparison<MembershipUserWrapper> comparison = null;
switch (sortDataBase)
{
    case "UserName":
        comparison = new Comparison<MembershipUserWrapper>(
            delegate(MembershipUserWrapper lhs, MembershipUserWrapper rhs)
            {
                return lhs.UserName.CompareTo(rhs.UserName);
            }
            );
        break;
    case "Email":
        comparison = new Comparison<MembershipUserWrapper>(
            delegate(MembershipUserWrapper lhs, MembershipUserWrapper rhs)
            {
                if (lhs.Email == null | rhs.Email == null)
                {
                    return 0;
                }
                else
                 return lhs.Email.CompareTo(rhs.Email);
            }
            );
        break;
    case "CreationDate":
        comparison = new Comparison<MembershipUserWrapper>(
            delegate(MembershipUserWrapper lhs, MembershipUserWrapper rhs)
            {
                return lhs.CreationDate.CompareTo(rhs.CreationDate);
            }
            );
        break;
    case "IsApproved":
        comparison = new Comparison<MembershipUserWrapper>(
            delegate(MembershipUserWrapper lhs, MembershipUserWrapper rhs)
            {
                return lhs.IsApproved.CompareTo(rhs.IsApproved);
            }
            );
        break;
    case "IsOnline":
        comparison = new Comparison<MembershipUserWrapper>(
            delegate(MembershipUserWrapper lhs, MembershipUserWrapper rhs)
            {
                return lhs.IsOnline.CompareTo(rhs.IsOnline);
            }
            );
        break;
    case "LastLoginDate":
        comparison = new Comparison<MembershipUserWrapper>(
            delegate(MembershipUserWrapper lhs, MembershipUserWrapper rhs)
            {
```

```
                        return lhs.LastLoginDate.CompareTo(rhs.LastLoginDate);
                    }
                    );
                break;
            default:
                comparison = new Comparison<MembershipUserWrapper>(
                    delegate(MembershipUserWrapper lhs, MembershipUserWrapper rhs)
                    {
                        return lhs.UserName.CompareTo(rhs.UserName);
                    }
                    );
                break;
        }
        if (sortData.EndsWith("DESC"))
        {
            memberList.Sort(comparison);
            memberList.Reverse();
        }
        else
            memberList.Sort(comparison);
    }
    return memberList;
}
```

Clase RoleDataObject.cs

En este caso, disponemos del archivo RoleDataObject.cs que contiene dos clases: RoleDataObject y RoleData.

La clase RoleDataObject es la que alberga los métodos para administrar los roles; se utiliza para encapsular el API de la configuración de roles o funciones. Por su parte la clase RoleData es un clase que únicamente tiene propiedades para almacenar los datos referentes a los Roles, es decir, algo similar a un objeto de negocio.

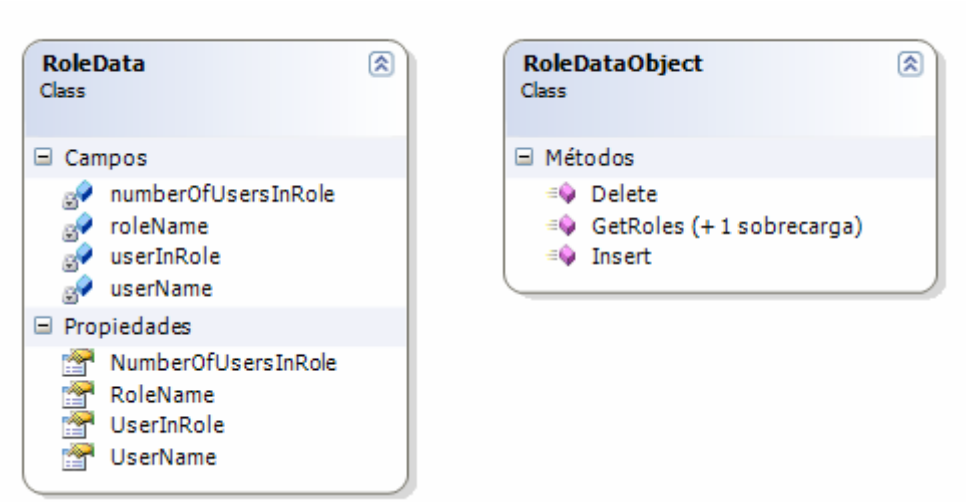


Fig. 11 Clase RoleDataObject.

Clase RoleDataObject

A continuación podemos observar en detalle los métodos que conforman esta clase.

Método Insert

Como es de suponer, a través del método Insert añadimos una nueva función al servicio de suscripciones.

```
[DataObjectMethod(DataObjectMethodType.Insert, true)]
    static public void Insert(string roleName)
    {
        if (Roles.RoleExists(roleName) == false)
        {
            Roles.CreateRole(roleName);
        }
    }
```

Método Delete

Como vemos, mediante este método podemos borrar cualquiera de los roles existentes.

```
[DataObjectMethod(DataObjectMethodType.Delete, true)]
    static public void Delete(string roleName)
    {
        MembershipUserCollection muc = Membership.GetAllUsers();
        string[] allUserNames = new string[1];
        foreach (MembershipUser mu in muc)
        {
```

```
            if (Roles.IsUserInRole(mu.UserName, roleName) == true)
            {
                allUserNames[0] = mu.UserName;
                Roles.RemoveUsersFromRole(allUserNames, roleName);
            }
        }
        Roles.DeleteRole(roleName);
    }
}
```

Método Select

El método asociado con la operación SELECT es GetRoles(), el cual está sobrecargado. De tal manera que podamos obtener todos las funciones existentes (GetRoles()) o aquellas que un determinado usuario tiene asignadas (GetRoles(string userName, bool showOnlyAssignedRolls)).

```
[DataObjectMethod(DataObjectMethodType.Select, true)]
    static public List<RoleData> GetRoles()
    {
        return GetRoles(null, false);
    }

[DataObjectMethod(DataObjectMethodType.Select, false)]
    static public List<RoleData> GetRoles(string userName, bool showOnlyAssignedRolls)
    {
        List<RoleData> roleList = new List<RoleData>();
        string[] roleListStr = Roles.GetAllRoles();
        foreach (string roleName in roleListStr)
        {
            bool userInRole = false;
            if (userName != null)
                userInRole = Roles.IsUserInRole(userName, roleName);
             if (showOnlyAssignedRolls == false || userInRole == true)
            {
                string[] usersInRole = Roles.GetUsersInRole(roleName);
                RoleData rd = new RoleData();
                rd.RoleName = roleName;
                rd.UserName = userName;
                rd.UserInRole = userInRole;
                rd.NumberOfUsersInRole = usersInRole.Length;
                roleList.Add(rd);
            }
        }
        return roleList;
    }
```

6. RSS

6.1. ¿Qué es RSS y para qué sirve?

RSS es un formato basado en XML que permite encontrar aquella información que mejor se adapta a lo que el usuario desea, pero también ofrecerla de forma rápida y actualizada.

Como sabemos, XML es un lenguaje de marcado extensible estricto de gran utilidad en el intercambio de datos, ya que permite describirlos sin mostrarlos al usuario, pero siendo a su vez legibles a través de diversas aplicaciones (navegadores, bases de datos, etc.)

Los archivos RSS son un nuevo método para obtener y ofrecer información gracias a que contienen metadatos sobre las fuentes de información. Este formato es de gran utilidad para sitios Web que actualicen sus contenidos con frecuencia, ya que permite compartir la información y verla en otros sitios de forma inmediata. A este intercambio de información se le denomina "sindicación".

La sindicación Web no es sólo un fenómeno vinculado a los weblogs, aunque han ayudado mucho a su popularización. Siempre se han sindicado contenidos y se ha compartido todo tipo de información en formato XML, de esta forma podemos ofrecer contenidos propios para que sean mostrados en otras páginas Web de forma integrada, lo que aumenta el valor de la página que muestra el contenido y también nos genera más valor, ya que normalmente la sindicación Web siempre enlaza con los contenidos originales.

El formato permite distribuir contenido sin necesidad de un navegador, utilizando un software diseñado para leer estos contenidos RSS. A pesar de eso, es posible utilizar el mismo navegador para ver los contenidos RSS. Las últimas versiones de los principales navegadores permiten leer los RSS sin necesidad de software adicional.

El software necesario para leer los feeds o canales RSS consiste en un programa llamado agregador o lector de feeds o canales, que es capaz de leer e interpretar dichas fuentes RSS. Gracias a estos programas se pueden obtener resúmenes de todos los sitios que deseemos desde el escritorio de nuestro sistema operativo, programas de correo electrónico o por medio

de aplicaciones Web que funcionan como tales. Con lo que, no es necesario abrir el navegador y visitar decenas de Webs para comprobar si en los múltiples sitios que ofrecen los contenidos que nos interesan se ha producido algún cambio o no.

El término anglosajón "feed" se utiliza para denominar a los documentos con formato RSS legibles por los agregadores o lectores de feeds.

Cualquier usuario puede suscribirse a un feed y obtener las últimas noticias enviadas a su lector RSS, el cual le alertará cuando haya nueva información para leer.

Éstos contienen un resumen de lo publicado en el sitio Web de origen. Se estructuran en uno o más ítems, donde cada ítem consta de un título, un resumen de texto y un enlace a la fuente original en la Web donde se encuentra el texto completo. Además puede incluir información adicional como el nombre del autor o la fecha y hora de publicación del contenido. Cualquier fuente de información capaz de poder ser fraccionada en ítems puede distribuirse utilizando RSS.

Algo importante a resaltar es que a partir de este formato se está desarrollando una cadena de valor nueva en el sector de los contenidos que está cambiando las formas de relación con la información tanto de los profesionales y empresas del sector como de los usuarios. Varias empresas están explorando nuevas formas de uso y distribución de la información.

Versiones de RSS

El acrónimo RSS se usa para referirse a los siguientes estándares:

- ⇒ Rich Site Summary (RSS 0.91)
- ⇒ RDF Site Summary (RSS 0.9 y 1.0)
- ⇒ Really Simple Syndication (RSS 2.0)
- ⇒ Además es usual que el término RSS sea usado indistintamente para referirse a cualquiera de los formatos RSS o Atom.

Hay que destacar que la versión 1.0 del lenguaje RSS también es conocida como RDF, por ello, algunos feeds o canales RSS están etiquetados como RSS 1.0 o RDF y guardados en archivos con extensión “.rdf”.

De la misma forma, a la versión 2.0 también se le llama RSS2. Por tanto algunos feeds o canales RSS están etiquetados como "RSS2" o "RSS 2.0".

Por otro lado, cabe mencionar la versión Atom. Ésta surge debido a los problemas de compatibilidad entre versiones y a algunas limitaciones adicionales de RSS. Viene descrita en un formato XML similar a RSS su objetivo es el mismo. Nació para resolver la confusión creada por la existencia de estándares similares para la sindicación (RSS y RDF) y para crear un API y un formato de redistribución más flexibles. La última versión del estándar es Atom 1.0 de julio de 2005. Las mejoras que supone respecto a RSS en cualquiera de sus versiones hacen que su uso se este extendiendo rápidamente a pesar de ser algo más complicado. Un documento Atom puede contener más información y es más consistente que un documento RSS, aunque dicha información es más compleja.

6.2. ¿Cómo se usa RSS?

Podemos usar el formato RSS en dos sentidos diferentes:

- ⇒ Para recibir información desde otros sitios Web
- ⇒ Para ofrecer información desde nuestra propia Web

Según seamos consumidores o creadores de contenidos elegiremos una u otra opción, aunque también podemos utilizar ambas a la vez.

Cómo usar el RSS para recibir información

Para poder utilizar el RSS y recibir contenidos, el usuario debe disponer de un agregador. Un lector o agregador de feeds es una aplicación local o basada en Web que interpreta los archivos RSS y visualiza su contenido.

Existe una gran variedad de lectores RSS, pero todos ellos se pueden clasificar en tres categorías:

- ⇒ **Agregadores de escritorio**: se instalan en el ordenador del usuario.
- ⇒ **Agregadores en línea**: no necesitan de instalación por parte del usuario. Suele bastar con darse de alta en el sitio del agregador para poder utilizarlo.

⇒ **Agregadores como plug-ins**: algunos navegadores y gestores de correo como Firefox, Nestcape, Opera, Thunderbird, etc. los incluyen en sus programas como servicio de valor añadido al usuario.

Una vez que el usuario dispone del agregador que haya elegido, debe seleccionar aquellos feeds o archivos RSS que sean de mayor interés para él y realizar la sindicación de contenidos.

Cada vez que el usuario añade un feed o canal a su programa agregador o lector de feeds se dice que se suscribe a ese feed. Podremos reconocer de forma sencilla de entre los diversos sitios e informaciones que existen en Internet aquellos que disponen de formato RSS, puesto que los feeds suelen indicarse en las páginas Web mediante pequeños cuadros que incluyen las siglas "RSS" o iconos como estos:

Sin embargo, no basta con pulsar sobre dichos iconos para ver la información que ofrecen los feeds, pues con ello aparece en el navegador del usuario una página en la que se puede ver el código del canal RSS. Sólo un agregador podrá interpretar ese código y mostrarlo de forma correcta al usuario para que pueda leer la información sin dificultad.

Para suscribirse a un feed, por lo general, el usuario debe copiar la dirección URL del archivo RSS y escribirla en su agregador.

Cómo usar el RSS para ofrecer información

Otro uso del RSS es el de ofrecer información desde nuestro sitio Web, pero ésta opción requiere de conocimientos previos sobre el lenguaje XML.

El código necesario para crear un feed o documento RSS debe incluir información sobre el sitio Web al que pertenece y que, por tanto, será información no variable, así como datos sobre los contenidos que ofrece y que se actualizarán cada breve periodo de tiempo. Esta será la información variable ofrecida en la sindicación.

Todos estos datos deben ir correctamente ordenados con sus correspondientes etiquetas de principio y final según lo establecido en el lenguaje de marcado XML. Así crearemos nuestro propio feed que puede contener varios artículos o ítems. Una vez creado el archivo RSS lo validamos para asegurarnos de que es correcto y lo registramos en varios agregadores para así comprobar cuántos usuarios se interesan por la información que les ofrecemos en nuestro feed.

6.3. ¿Cómo crear RSS?

El formato RSS está basado en XML, por lo que para cualquiera de las tres especificaciones existentes sería recomendable tener unas nociones básicas de este lenguaje de marcado.

Un documento XML debe incluir una declaración que lo determine como tal. Por eso, la primera línea de nuestro código RSS será la que define el lenguaje de marcado y el tipo de caracteres que vamos a utilizar.

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
```

A continuación debemos escoger alguna de las especificaciones de RSS que existen. En nuestro ejemplo seguiremos las reglas de sintaxis de "Really Simple Syndication (RSS 2.0)", porque -como su propio nombre indica- se construye sobre un conjunto de normas bastante sencillas, aunque estrictas y es la más extendida. Así que la siguiente línea de nuestro código indicará que seguimos estas reglas:

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
        <rss version="2.0">
</rss>
```

Todos las demás etiquetas que vamos a utilizar para crear el feed deben situarse entre estas dos de <rss>, porque son las que indican que estamos creando un canal RSS.

A continuación, debemos crear un "canal" en el que introduciremos los contenidos que queremos mostrar a los demás usuarios. Bastará con escribir dos etiquetas <channel> -una de principio y otra de final- a continuación de lo que ya llevamos hecho:

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
        <rss version="2.0">
                <channel>
                </channel>
        </rss>
```

Por otro lado, cabe mencionar que todo feed o canal RSS se compone de dos partes esencialmente:

- ⇒ Elementos no variables
- ⇒ Elementos variables

Existen distintos elementos no variables, pero sólo tres son obligatorios:

- ⇒ Título: <title>
- ⇒ Enlace: <link>
- ⇒ Descripción: <description>

Estos tres elementos deben describir nuestro canal RSS en general, por lo que el Título hará referencia al nombre del feed, el Enlace será la URL de nuestro sitio Web y la Descripción informará al usuario del tipo de contenidos que vamos a incluir en el canal.
Estas tres líneas de código se escriben entre las dos etiquetas <channel> que hemos ya creado:

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
        <rss version="2.0">
                <channel>
                        <title>El nombre de nuestro feed</title>
                        <link>Dirección Web en la que se encuentre nuestro RSS</link>
                        <description>Contenido que vas a ofrecer a los usuarios </description>
                </channel>
        </rss>
```

Los elementos variables de un canal RSS se denominan "ítem" y pueden incluirse varios en un mismo canal.

De nuevo, cada ítem lo creamos con una etiqueta de principio y otra de final. Se sitúan entre las de <channel> justo después de los elementos no variables. Obligatoriamente debemos incluir en nuestro canal tres elementos variables, aunque, como en el caso anterior, existen más.

Estos elementos obligatorios vuelven a ser:

- ⇒ Título: <title>
- ⇒ Enlace: <link>
- ⇒ Descripción: <description>

Pero ahora estos elementos describen cada uno de los artículos o informaciones que vamos a ofrecer y cuyo contenido iremos actualizando cada cierto tiempo.

El código para incluir un ítem en nuestro canal RSS sería el siguiente:

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
        <rss version="2.0">
                <channel>
                        <title>El nombre de nuestro feed</title>
                        <link>Dirección Web en la que se encuentre nuestro RSS</link>
                        <description>Contenido que vas a ofrecer a los usuarios </description>
                        <item>
                                <title>Título del artículo</title>
                                <link>Dirección Web a la que podemos ir para ampliar esta
                                información</link>
                                <description>Contenido de esta información</description>
                        </item>
                </channel>
        </rss>
```

Por último, y para hacer accesible el canal RSS, debemos poner un enlace a él desde nuestro sitio Web. Bastará con que insertemos una línea como la que se muestra a continuación en el código HTML de la página en la que quieras que aparezca el enlace:

```
<a type="application/rss+xml" href="nombre del canal RSS.xml">
<img src="imagen que queremos que sirva de enlace.gif">
</a>
```

7. AJAX y ASP .NET AJAX

7.1. AJAX

7.1.1. Introducción

Hasta hace relativamente poco, el interfaz de usuario de las aplicaciones Web resultaba muy pobre y poco atractivo. Debido a las restricciones y problemas de compatibilidad entre navegadores, los desarrolladores Web se veían obligados a recortar la funcionalidad de sus páginas, lo que obligaba a que el peso de la aplicación recayera sobre el lado del servidor. Por otro lado, la existencia de varios tipos de navegadores y la falta de acuerdo a la hora de establecer o acogerse a estándares ha ocasionado que muchos avances tecnológicos no se hayan puesto en práctica hasta hace muy poco.

En la actualidad, parece que esta situación ha mejorado y que comienza a promoverse cierto consenso en este sentido, ya que casi todos los navegadores modernos incluyen tecnologías que posibilitan al diseñador Web manipular los controles de las páginas de forma dinámica, así como tecnologías que permiten encapsular información y comunicarse con el servidor sin tener que hacer un POST de la página completa con su consecuente refresco.

La unión de varias tecnologías que ya existentes (DOM, DHTML, XMLHttpRequest,…), permite que el usuario pueda trabajar en la página mientras que paralelamente se realiza una operación de petición de datos/validación a servidor (mediante XMLHttp). Una vez que se recibe la respuesta, usando DHTML actualizamos el fragmento de la página que cambia.

El conjunto de estas tecnologías que nos permiten establecer comunicación entre cliente y servidor, así como realizar un interfaz de usuario avanzado y que hacen que las aplicaciones Web se acerquen cada vez más a lo que conocemos como aplicaciones desktop o de escritorio, recibe el nombre de AJAX, Asynchronous JavaScript and XML.

En esencia, AJAX pretende ser una técnica de desarrollo Web para crear aplicaciones interactivas que se ejecutan en el cliente y que mantienen comunicación asíncrona con el

servidor en segundo plano. De esta forma es posible realizar cambios sobre la página sin necesidad de recargarla, lo que aumenta la interactividad, velocidad y usabilidad de la misma.

7.1.2. Tecnologías que conforman AJAX

Las tecnologías más relevantes que conforman AJAX son:

- ⇒ **XML y XSLT** – para el intercambio y manipulación de datos. XML, acrónimo en inglés de Extensible Markup Language (lenguaje de marcado extensible) es un metalenguaje que nos permite definir lenguajes de marcado adecuados para diferentes necesidades. Es el estándar usado comúnmente para la transferencia de datos con el servidor. Por otro lado XSLT (XML Stylesheets Language for Transformation), o lenguaje de transformación basado en hojas de estilo, representa un estándar que presenta una forma de transformar documentos XML en otros documentos, particularmente otros documentos XML. Son hojas de estilo que realizan la transformación del documento utilizando una o varias reglas de plantilla. La unión de XML y XSLT permite separar contenido y presentación, aumentando así la productividad.
- ⇒ **XHTML y CSS** – para estructurar y presentar la información basándose en estándares. XHTML es el acrónimo inglés de Extensible Hypertext Markup Language (lenguaje extensible de marcado de hipertexto) y es el lenguaje de marcado pensado para sustituir a HTML como estándar para las páginas Web. XHTML es la versión XML de HTML, por lo que tiene, básicamente, las mismas funcionalidades, pero cumple las especificaciones, más estrictas, de XML. CSS es el acrónimo inglés de Cascade Style Sheets (hojas de estilo en cascada). Las hojas de estilo permiten organizar la presentación de una página y separan el contenido de la forma de presentarlo.
- ⇒ **Document Object Model (DOM)** – para la visualización e interacción dinámica. El objeto DOM (Modelo de Objetos de Documentos), define la estructura lógica de los documentos y el modo en que se accede y manipulan los mismos. Es una interfaz de programación de aplicaciones (API) para acceder, añadir y cambiar dinámicamente contenido estructurado con leguajes como JavaScript. En resumen, permite al navegador interactuar de manera dinámica con los datos.

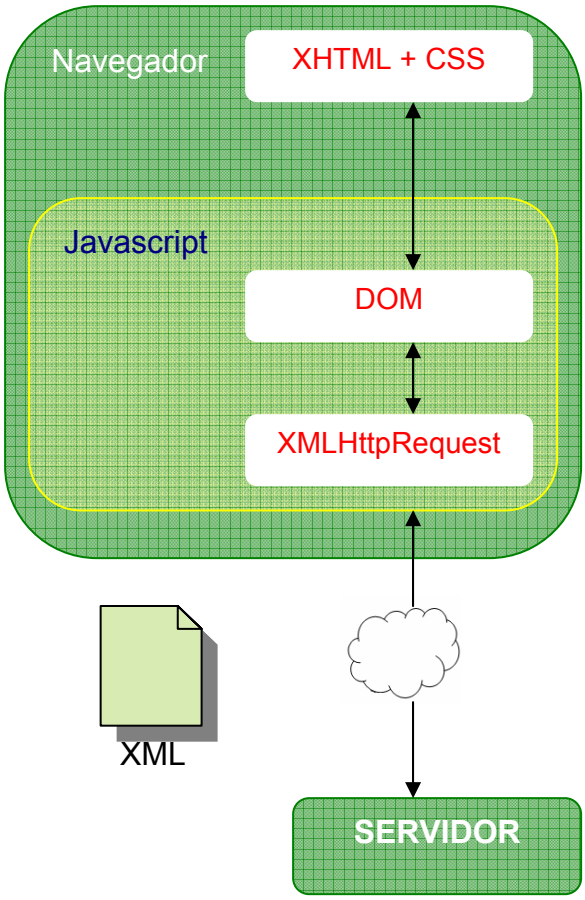


Fig. 12 Tecnologías que conforman AJAX.

⇒ **XmlHttpRequest** – para la recuperación de datos asíncrona / comunicación asíncrona con el servidor. XmlHttpRequest es un objeto que permite al navegador Web realizar peticiones HTTP y HTTPs a un servidor remoto sin la necesidad de hacer un post de toda la página Web, es decir de manera asíncrona. Se utiliza para transferir y manipular datos XML hacia y desde el navegador Web, estableciéndose un canal de conexión independiente entre el lado del cliente de la página Web y el servidor. En esencia, se pueden realizar y recibir peticiones HTTP en segundo plano y sin que el usuario experimente ninguna interrupción visual. En Internet Explorer está disponible con el componente MSXML ActiveX. Mientras que con Mozilla Firefox y otros navegadores, esta característica es proporcionada por un objeto llamado literalmente XmlHttpRequest. El objeto XmlHttpRequest esta modelado después que el componente MSXML. Las bibliotecas JavaScript en el lado del cliente ocultan las diferencias entre los diferentes

entornos de los navegadores. A veces estas comunicaciones se realizan a través de un objeto IFrame.

⇒ **JavaScript –** es la tecnología usada para unir estas tecnologías / enlazarlo todo. JavaScript es un lenguaje de script interpretado. Fue diseñado para agregar interactividad a las páginas Web y normalmente es integrado dentro de las mismas. Se ejecuta en el navegador al mismo tiempo que las sentencias van descargándose junto con el código HTML. Es frecuentemente utilizado para capturar eventos de la página Web, tales como: finalización de la carga de la página, el usuario presionó un botón, etc. También es usado para validar datos de la página Web y cambiar el contenido de los elementos HTML de la misma. La versión de JavaScript debe ser la 1.5 o posterior. Aunque JavaScript no se requiere de manea especifica, es necesario desde el punto de vista de que es el único lenguaje de script soportado por la mayoría de los navegadores actuales.

Como vemos, todas estas tecnologías no son nuevas, lo que resulta novedoso es la forma de utilizarlas.

7.1.3. Ejecutar aplicaciones AJAX

Desafortunadamente no todos los navegadores soportan Ajax. Para ejecutar Ajax, un navegador debe cumplir los siguientes requisitos:

⇒ Ser relativamente moderno. Las aplicaciones que utilizan Ajax no están disponibles en todas las versiones de todos los navegadores. Mientras que las últimas versiones de Internet Explorer, Firefox u Opera dan soporte a este tipo de aplicaciones, sus versiones anteriores pueden ser problemáticas.

⇒ Soportar DOM.

⇒ Utilizar JavaScript.

⇒ Soportar XML y XSLT.

⇒ Posibilitar la activación de ActiveX en el cliente. En el caso de Internet Explorer podemos tener problemas si no tenemos activado ActiveX.

Navegadores que soportan AJAX

- ⇒ Navegadores basados en Gecko como Mozilla, Mozilla Firefox, SeaMonkey, Camino, K-Meleon, Flock, Epiphany, Galeon y Netscape versión 7.1 y superiores.
- ⇒ Microsoft Internet Explorer para Windows versión 5.0 y superiores, y los navegadores basados en él.
- ⇒ Opera versión 8.0 y superiores, incluyendo Opera Mobile Browser versión 8.0 y superiores.

Navegadores que no permiten AJAX

- ⇒ Opera 7 y anteriores.
- ⇒ Microsoft Internet Explorer para Windows versión 4.0 y anteriores.
- ⇒ Microsoft Internet Explorer para Macintosh, todas las versiones.
- ⇒ Dillo.
- ⇒ Navegadores basados en texto como Lynx y Links.
- ⇒ Navegadores para incapacitados visuales (braille).

7.1.4. Ventajas e inconvenientes de AJAX

Sus principales ventajas son:

- ⇒ Asíncrono – Permite realizar llamadas asíncronas a un servidor Web. Lo que hace que el navegador del cliente no tenga que esperar la llegada de todos los datos antes de permitir actuar al usuario una vez más.
- ⇒ Transferencia de datos mínima y mayor rapidez en las operaciones – Al no realizar un post completo ni enviar todos los datos al servidor, la utilización de la red se minimiza y las operaciones se realizan con mayor rapidez.
- ⇒ Menos carga del servidor – Dado que únicamente enviamos los datos necesarios al servidor, este no tiene que procesar todos los elementos.
- ⇒ Contexto – Con un post completo los usuarios pueden perder el contexto donde están. Si el usuario se encuentra en la parte inferior de la página y presiona el botón de envío será redirigido a la parte superior de la página. Con Ajax no hay post completo por lo que el usuario mantiene su localización. El estado del usuario se mantiene y éste ya no tiene que desplazarse hacia abajo para localizar donde se encontraba antes de presionar el botón de envío.

- ⇒ Más usabilidad – Permite trabajar mientras la operación contra el servidor se procesa en paralelo.
- ⇒ Más cerca de crear realmente "Aplicaciones Web".
- ⇒ Menos ancho de banda (permite el ahorro de mucho dinero si tenemos muchas visitas).
- ⇒ Elimina recarga completa de páginas a favor de actualizaciones incrementales y más reducidas
- ⇒ Incrementa el procesamiento en el lado del cliente haciendo que el navegador sea más responsable en ciertos aspectos de la ejecución.
- ⇒ Explota las actuales capacidades gráficas de los navegadores para añadir más interactividad

Mientras que los principales inconvenientes son los siguientes:

- ⇒ Deja de existir el botón atrás y el botón adelante, los usuarios deben cambiar su manera de entender los sitios Web. En una aplicación Web clásica, el usuario puede, mediante el navegador, ir hacia la página anterior o a la siguiente —si es que las hay— del historial. En Ajax, eventualmente, no se cambiará de página (sólo se modificará parte de su contenido, de acuerdo con cada proceso) por lo que las teclas atrás y adelante no funcionarán y habrá que buscar formas alternativas para moverse (modificando el historial con JavaScript o utilizando iframes invisibles para generarlo) dentro de la aplicación Web.
- ⇒ El usuario no está acostumbrado (espera ver una carga de página).
- ⇒ Normalmente, el tiempo muerto mientras se espera que el servidor retorne una respuesta es notorio por aspectos como la pantalla en blanco o señales dadas por el navegador (descargando desde). En una aplicación Ajax, las peticiones se hacen en segundo plano, por lo que habrá que "avisar" (visualmente) al usuario que recargamos alguna parte de la página, mediante barras de carga propia o elementos animados. Sería necesario colocar estas alertas visuales en un lugar privilegiado.
- ⇒ Problemas con los buscadores, ya que no todos son compatibles.
- ⇒ No podemos pasar URLS de partes concretas de la Web (dejan de existir en parte los famosos Permalinks)
- ⇒ Guardar documentos: al generar contenido de forma dinámica en el lado cliente, si se almacena una página puede que su contenido (al visualizarse offline) no refleje lo que el usuario estaba viendo al momento de guardarla en disco. En el mismo sentido, surge el problema de agregar una página particular a la lista de favoritos.

- ⇒ No podemos obtener la url concreta de donde estamos.
- ⇒ Dependiendo de la carga del servidor podemos experimentar tiempos tardíos de respuesta.
- ⇒ Necesario JavaScript. Uno de los basamentos de Ajax se ubica en este lenguaje. Si bien la mayoría de los navegadores actuales lo soporta, no se puede dar por cierto que todos los usuarios utilicen navegadores actuales. También es posible deshabilitar el soporte completo —o determinadas opciones— de JavaScript, lo que atentaría contra el correcto funcionamiento de las aplicaciones basadas en Ajax. En el caso de Internet Explorer 6, será necesario tener activado el soporte para objetos ActiveX ya que la implementación de XMLHttpRequest así lo requiere.

7.1.5. AJAX contra Aplicaciones Web tradicionales – ¿cómo funciona?

En el modelo clásico de aplicaciones Web la mayoría de las acciones de los usuarios en el interfaz producen una petición HTTP al servidor Web. El servidor al recibir dicha petición realiza diferentes procesos en función de la misma y a continuación contesta retornando una nueva página Web al cliente. Esta manera de proceder desperdicia mucho ancho de banda, ya que parte del HTML enviado en la segunda página, ya estaba presente en la primera. Mientras tanto, y al otro lado de la comunicación, el usuario no hace más que esperar.

Es obvio que como desarrolladores no queremos hacer esperar a los usuarios. Una vez que la interfaz esta cargada, no tenemos por qué detener la interacción del usuario cada vez que la aplicación necesita algo del servidor.
Una aplicación AJAX elimina la naturaleza “start-stop-start-stop”, característica de la interacción Web de las aplicaciones tradicionales. Para ello, el modelo AJAX introduce una capa intermedia entre el usuario y el servidor. Este intermediario se denomina motor AJAX (AJAX engine) y es en realidad un objeto o función JavaScript que acompaña al HTML. El motor se carga en el primer acceso que se hace desde el navegador a la aplicación Web y tiene una doble responsabilidad: primero, generar y actualizar la interfaz visualizada por el usuario y segundo, comunicarse con el servidor en representación del usuario. Esta comunicación ocurre de manera asíncrona evitando que el usuario vea una página en blanco o el típico reloj de arena cada vez que realice una acción.

En el primer acceso que se hace desde el navegador a la aplicación Web se baja gran parte del código JavaScript, y en muchos casos, una parte muy pequeña del HTML de la interfaz.

Ahora cada acción de un usuario, que normalmente generaría una petición HTTP, es controlada por el motor AJAX a través de JavaScript. Éste captura los eventos del usuario (clicks, drag-n-drop, etc.), si la acción que implican no requiere un viaje al servidor (por ejemplo: validación de datos, edición de datos en memoria e incluso la propia navegación) la procesa él mismo; en caso contrario (por ejemplo: una búsqueda en la base de datos) envía una petición en segundo plano y de manera asíncrona por HTTP a la parte servidora utilizando el objeto XMLHttpRequest. El hecho de que la petición sea asíncrona hace que el código que se está ejecutando no espere una respuesta antes de continuar lo que conlleva que el navegador no se quede “bloqueado” y el usuario pueda seguir interactuando con la aplicación sin esperar a recibir la respuesta. Sin embargo, en muchas ocasiones esta ventaja no se explota, ya que o la respuesta llega rápido o no tiene sentido para el usuario hacer algo mientras tanto.

Con el nuevo modelo, los vínculos, en vez de proporcionar enlaces a otros recursos (como otras paginas Web), realizan una llamada al motor de AJAX, el cual programa y ejecuta la petición.

El servidor, que tradicionalmente presenta HTML, imágenes, CSS o JavaScript, está configurado para devolver datos que el motor de AJAX pueda usar. Estos datos pueden ser texto plano, XML o cualquier otro formato de datos que podamos necesitar. El único requisito es que el motor pueda entender e interpretar estos datos. Cuando el motor de AJAX recibe una respuesta del servidor, entra en acción analizando los datos y realizando diversos cambios en el interfaz de usuario basándose en la información que se le proporciona mediante DHTML y DOM. Debido que este proceso implica transferir mucha menos información que el modelo clásico de aplicaciones Web, la actualización de la interfaz de usuario es más rápida y el usuario puede hacer su trabajo más ágilmente.

La parte servidora, la igual que en el modelo clásico de aplicaciones Web, recibe peticiones HTTP, invoca la lógica de negocio y devuelve datos al cliente.

En las siguientes figuras podemos ver la diferencia entre el modelo tradicional y el de AJAX.

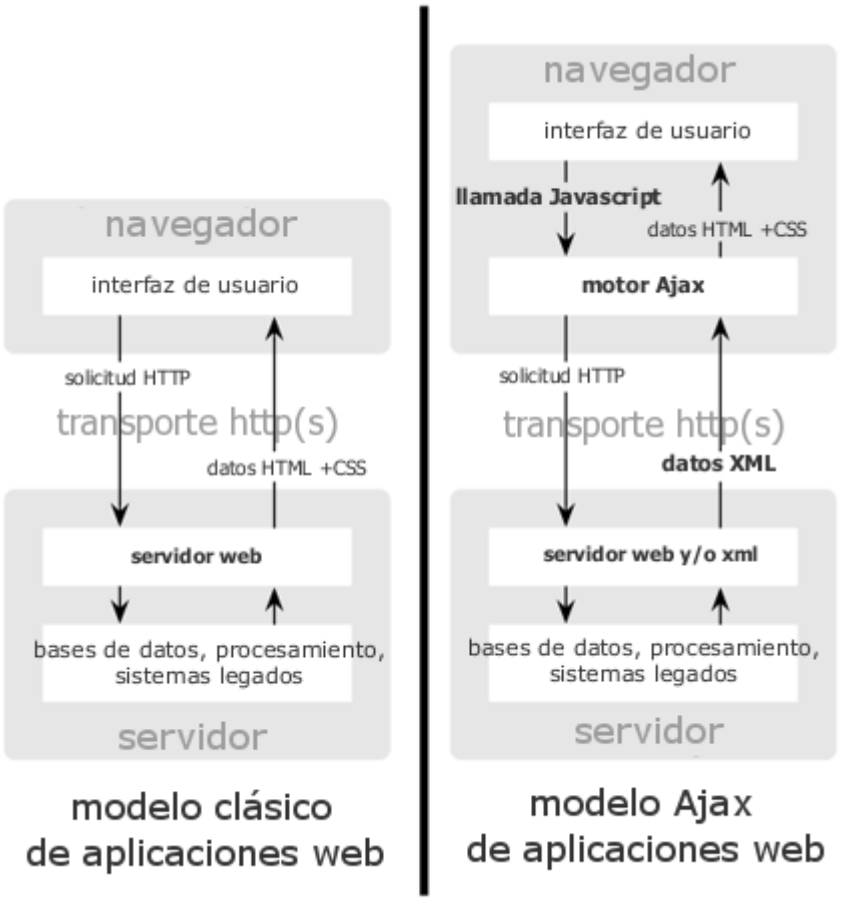


Fig. 13 Comparación del esquema de una aplicación Web Tradicional y el de una con AJAX.

7.1.6. Limitaciones

Como hemos visto, la tecnología AJAX nos ofrece grandes ventajas para el desarrollo de nuestras aplicaciones Web, pero también tiene sus limitaciones:

- ⇒ El desarrollo de aplicaciones con AJAX requiere del conocimiento de los objetos del documento (DOMs), los cuales difieren entre los distintos navegadores.
- ⇒ JavaScript no ofrece todas las características de la programación orientada a objetos, ni del código administrado que el.NET Framework ofrece.
- ⇒ Crear aplicaciones Web, con programación del lado del cliente requiere del uso de un nuevo lenguaje y una nueva plataforma de desarrollo.

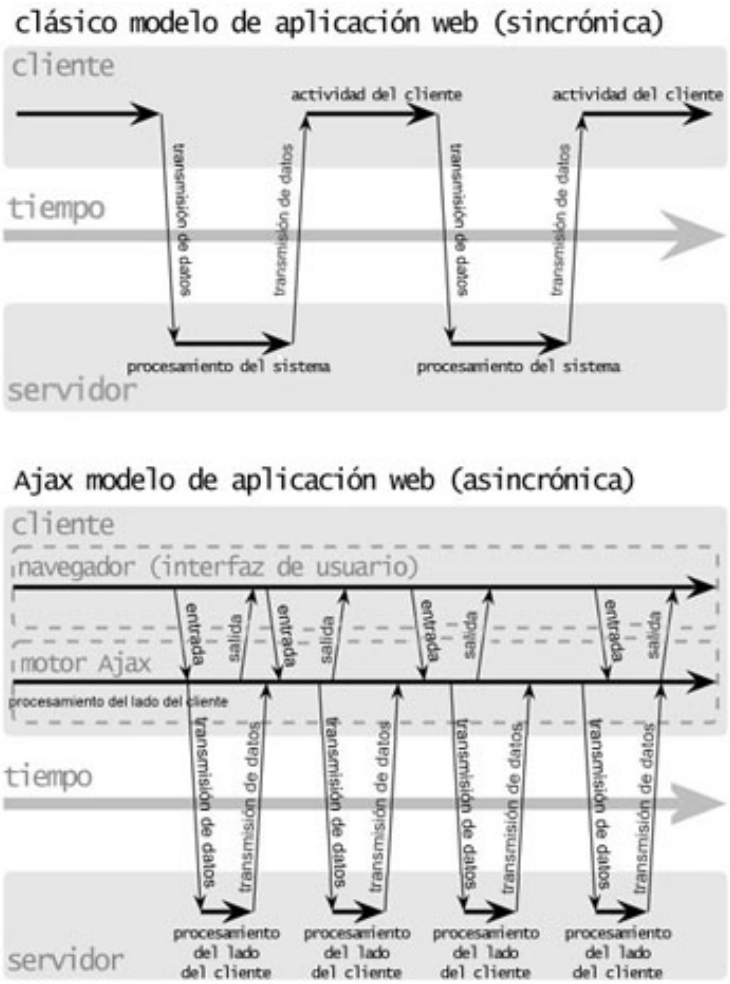


Fig. 14 Comparación entre el modo síncrono de una aplicación Web Tradicional y el modo asíncrono con AJAX.

7.2. ASP .NET AJAX

7.2.1. Introducción

ASP.NET AJAX es una nueva tecnología de Microsoft para el desarrollo Web que facilita el uso y extiende los beneficios de la metodología AJAX sobre la plataforma ASP .NET.

Quizá su principal beneficio es que todo se puede hacer visualmente sin necesidad de escribir una sola línea de código en Javascript.

ASP.NET AJAX integra bibliotecas de script de cliente con el framework de desarrollo de aplicaciones Web ASP.NET 2.0, brindando una gran facilidad para tener en nuestras aplicaciones Web las ventajas que nos ofrece el uso de AJAX.

Al estar integrado con el framework de ASP.NET 2.0, además de facilitar el uso de técnicas AJAX ofreciéndonos la misma plataforma de desarrollo del lado del cliente, ASP.NET AJAX va más allá, extendiendo los beneficios que este último nos ofrece, permitiendo a los desarrolladores la creación de páginas Web de manera rápida y con una enriquecida e interactiva interfaz de usuario con una comunicación con el servidor más eficiente, con una comunicación más eficiente cliente-servidor, añadiendo simplemente unos pocos controles de servidor en nuestras páginas.

El framework de ASP.NET Ajax se basa en cuatro partes bien diferenciadas:

- ⇒ **ASP.NET AJAX Extensions:** El núcleo de la implementación de AJAX. Proporciona un pequeño núcleo de controles de servidor que permiten realizar la mayor parte de las operaciones cliente / servidor.
- ⇒ **ASP.NET AJAX Library:** Es un conjunto de bibliotecas y de clases Javascript que facilitan el trabajo del desarrollador en el navegador cliente. Además, facilitan una serie de métodos y patrones para realizar llamadas a servidor directamente desde Javascript. Estas bibliotecas se distribuyen dentro de ASP.NET AJAX Extensions.
- ⇒ **ASP.NET AJAX Control Toolkit:** Una serie de controles que, basándose en las extensiones AJAX, expanden la funcionalidad de los controles y proveen al desarrollador de controles ricos que pueden usar en la aplicación (por ejemplo, un control de selección de fecha, un control que permite tener múltiples pestañas en la aplicación, etc.). Este toolkit se ha creado y se mantiene entre Microsoft y la comunidad de codeplex, y por tanto, es de código abierto y cualquiera que quiera contribuir con el desarrollo puede hacerlo.
- ⇒ **ASP.NET AJAX Futures:** Nuevas funcionalidades que se incluirán en subsiguientes implementaciones del framework.

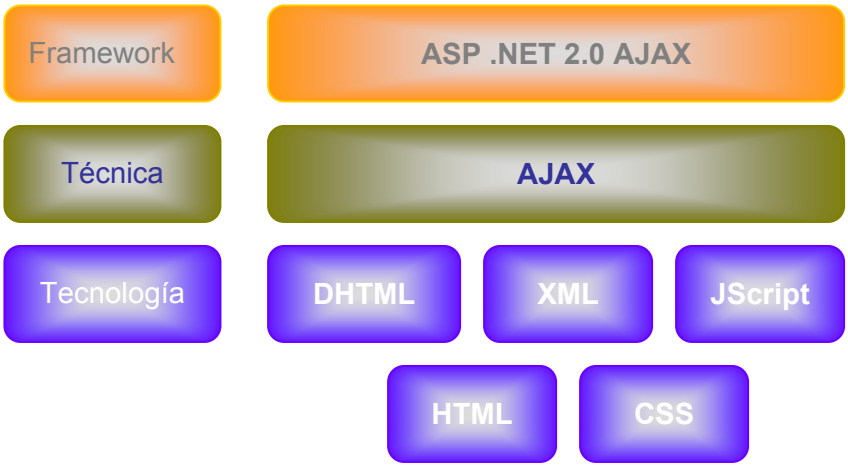


Fig. 15 ASP .NET 2.0 + AJAX.

7.2.2. Arquitectura de ASP .NET AJAX

ASP.NET AJAX consiste en una serie de bibliotecas de scripts en el lado del cliente y componentes en el lado del servidor que están integrados con el fin de proporcionar una plataforma de desarrollo robusta.

En la **Fig. 16** observe la funcionalidad de la biblioteca en el lado del cliente de Microsoft AJAX, la cual nos provee de un conjunto de soluciones para crear aplicaciones Web basadas en el cliente, con soporte a desarrollo orientado a objetos, compatibilidad entre navegadores, llamadas a servicios Web services de manera asíncrona, además de componentes para la creación de ricas interfaces de usuario, y los servicios se conexión de redes y núcleo. Además muestra la funcionalidad de las extensiones de servidor (ASP.NET 2.0 AJAX Extensions), las cuales incluyen un soporte para los scripts, los servicios Web, los servicios de la aplicación y los controles de servidor.

Fig. 16 Arquitectura de ASP .NET AJAX.

Modelo Basado en el Cliente

Las características para el desarrollo orientado a objetos incluidas en las bibliotecas de código cliente de ASP.NET aportan un alto nivel de consistencia y modularidad. Las siguientes capas están incluidas dentro de las bibliotecas de código de ASP.NET AJAX.

Para el desarrollo en cliente ASP.NET AJAX contiene unas bibliotecas de código cliente que consisten en archivos Javascript (.js), que definen una capa que nos permite el acercamiento al desarrollo de aplicaciones basadas en cliente y que a su vez proporcionan las características necesarias para el desarrollo orientado a objetos.

Esta capa consiste en:

- ⇒ Una capa para la compatibilidad de navegadores, permitiendo que nuestro código ASP.NET AJAX funcione de manera adecuada en la mayoría de los navegadores eliminando la necesidad de escribir código específico para cada navegador.
- ⇒ Conjunto de servicios ASP.AJAX el cual agrega extensiones a JavaScript tales como clases, espacios de nombres, manejo de eventos, tipos de datos, herencia, y

serialización de objetos. Como vemos, estas son características de la programación orientada a objetos por lo que esta nueva tecnología nos brinda un modelo de programación muy familiar.

⇒ Una biblioteca de clases base, la cual incluye componentes tales como string builders, debuggers, timers, y tracing.

⇒ Una capa de red para comunicarse con servicios Web, aplicaciones y administrar de forma asíncrona llamadas a métodos remotos. Esta capa administra la complejidad de realizar llamadas asíncronas mediante XMLHTTP, reduciendo esto a unas cuantas líneas de código.

⇒ Una capa para la interfaz de usuario, la cual brinda de ciertas capacidades al cliente tales como: comportamientos, la sintaxis declarativa de ASP.NET AJAX que permite crear componentes ASP.NET AJAX de la misma forma que se crean controles ASP.Net., componentes UI, y conexión con datos.

⇒ Una capa que crea controles específicos de ASP.NET AJAX para el desarrollo en cliente. Estos controles pueden estar ligados a datos o a comportamientos como el drag and drop, y otros más.

Dada las características de la biblioteca cliente de ASP.NET AJAX, podemos conceptualizarla como un subconjunto de la arquitectura de ASP.NET, utilizando todo el poder del script y Html dinámico (DHTML) enriqueciendo y volviendo más interactiva la experiencia del usuario tales como las aplicaciones mash-ups, gadgets entre otras.

Modelo basado en el servidor

La parte de servidor de ASP.NET AJAX consiste en una serie de componentes de servidor, servicios, y controles que pueden ser integrados con código cliente ASP.NET AJAX. Estos proporcionan un control sobre la interfaz de usuario y el flujo de la aplicación además de controlar la serialización, la validación, la extensibilidad de los controles, etc. También nos encontramos con la posibilidad de utilizar servicios Web que hacen posible lograr funcionalidades de ASP.NET tales como: perfiles, membership, roles, personalización, y la globalización y servicios de cultura específica.

Dentro de estos componentes tenemos:

- ⇒ Servicios Web que hacen posible lograr funcionalidades de ASP.NET tales como: perfiles, membership, roles, personalización, y la globalización y servicios de cultura específica.
- ⇒ Controles de servidor ASP.NET AJAX los cuales reensamblan controles de servidor ASP.NET pero que generan código cliente ASP.NET AJAX. El comportamiento de estos controles de servidor ASP.NET AJAX se asemeja mucho al de los controles ASP.NET tales como los botones, etiquetas, opciones, cajas de texto, check boxes, hyperlinks, y controles de validación. Aquellos que estén familiarizados con el uso de los controles ASP.NET los controles de servidor ASP.NET AJAX les serán de gran utilidad para no generar manualmente código cliente.
- ⇒ Controles de servidor ASP.NET AJAX que generan JavaScript para producir comportamientos en cliente, entre estos controles contamos con el HoverBehavior control, ClickBehavior control, Popup control, y el AutocompleteBehavior control.

Todos los controles ASP.NET AJAX pueden estar integrados con Visual Studio por lo que podemos trabajar con ellos en el diseñador de la herramienta tal como lo hacemos con los controles estándar de ASP.NET.

Por lo tanto con el desarrollo Web con Ajax centrado en el servidor logramos enriquecer aplicaciones sin los bloques de código JavaScript requeridos permitiendo mantener toda la lógica interface de la aplicación en el servidor.

7.2.3. ASP.NET AJAX SERVER CONTROLS (Controles del lado del servidor)

Los controles ASP.NET AJAX server consisten en código tanto en el lado del cliente como en el del servidor que se integra para producir el comportamiento AJAX. Los principales controles ASP.NET AJAX del lado del servidor son los siguientes:

- ⇒ **ScriptManager:** este control proporciona el mecanismo para añadir soporte AJAX a una página .aspx. Se encarga de controlar los componentes ASP.NET AJAX, el trazado parcial de páginas (partial page rendering), las peticiones del cliente, y las respuestas del servidor en páginas Web ASP.NET.
- ⇒ **UpdatePanel:** permite refrescar partes seleccionadas de una página. A la hora de actualizar, en lugar de realizar un post de la página completa al servidor, actualiza

únicamente la parte afectada. Especifica las regiones que van a ser actualizadas en el trazado parcial de la página.

⇒ **UpdateProgress:** su funcionalidad es la de mostrar un mensaje mientras una petición viaja al servidor y éste responde. Proporciona información acerca de las actualizaciones en los UpdatePanel.

⇒ **Timer:** realiza postbacks en intervalos de tiempo definidos. Podemos usarlos para enviar la página completa o combinarlo con el control UpdatePanel. Esto último nos da la posibilidad de refrescar fragmentos de páginas con distintas temporizaciones, dependiendo de la prioridad de los datos.

⇒ **ScriptManagerProxy:** se emplea en caso de que el ScriptManager se encuentre en otra página, por ejemplo, en una página maestra, y se necesite acceder a las propiedades del mismo ScriptManager. Por ejemplo, incluir un javascript en una página y que el ScriptManager de la página maestra lo cargue en el cliente.

ScriptManager

Es uno de los principales controles del framework. Es el que provee a la página de la funcionalidad de cliente, como por ejemplo, realizar las llamadas asíncronas al servidor, realizar refrescos parciales de páginas, etc.

Es necesario para realizar cualquier operación con el framework, por lo tanto imprescindible para usar cualquier control AJAX, tanto los controles UpdatePanel, UpdateProgress y Timer, como los controles del toolkit. Cada página que vaya hacer uso de la tecnología AJAX tendrá un ScriptManager. Sólo deberemos añadir una única instancia de ScriptManager en cada página, por lo que si trabajamos con MasterPages, dicha instancia la añadiremos en la página maestra. Si queremos añadir una funcionalidad adicional de cliente (scripts, servicios, etc.…), específica en cada página, contamos con el control ScriptManagerProxy. Por otro lado es importante especificar el atributo EnablePartialRendering (Ver **Fig. 17)**; ya que es él que nos permitirá realizar modificaciones parciales de las páginas.

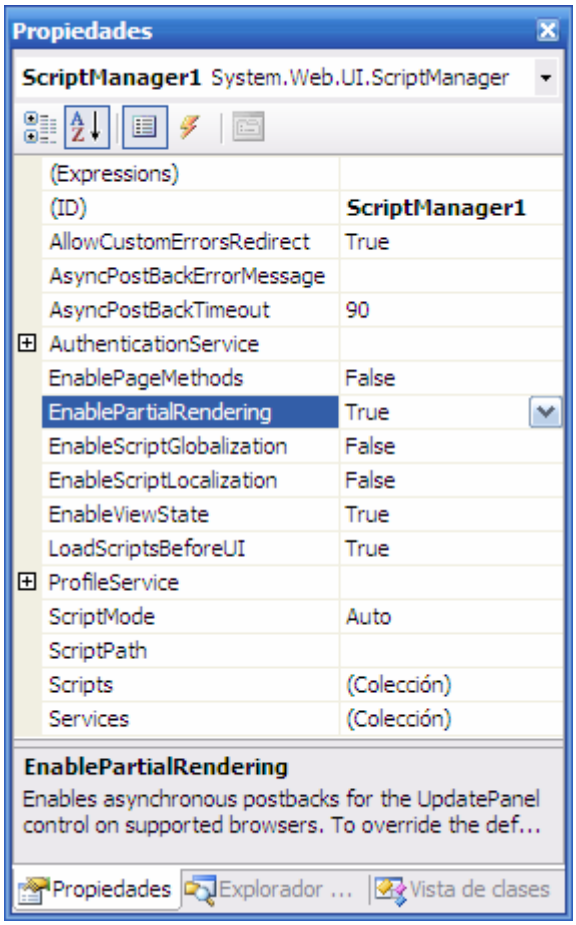


Fig. 17 ScriptManager

UpdatePanel

En una aplicación Web tradicional cualquier actualización o validación de una página que necesite los servicios del servidor implica tener que hacer un post a servidor de la página completa, dejar ésta inoperante hasta que se recibe la nueva página, y recargarla completamente en el navegador del cliente. Para evitar esto la plataforma ASP.NET AJAX nos ofrece el control UpdatePanel. Poniendo un par de etiquetas en nuestra página ASP.NET delimitamos las áreas que se tienen que actualizar de forma asíncrona, y el motor de AJAX se encarga de generar todas las llamadas fuera de banda y las actualizaciones en el navegador.

Las operaciones que realiza este control son:

- Intercepta cada comando que provoca un postback al servidor.
- Lo desvía a una llamada XMLHTTP en la que se manda el ViewState de la aplicación.
- Envía de vuelta sólo el fragmento de página que ha modificado.

- Actualiza la página en cliente mediante DOM.

De esta forma la página genera menos tráfico ya que no se tiene que enviar completa en un post de ida y vuelta. La **Fig. 18** muestra las propiedades de este control:

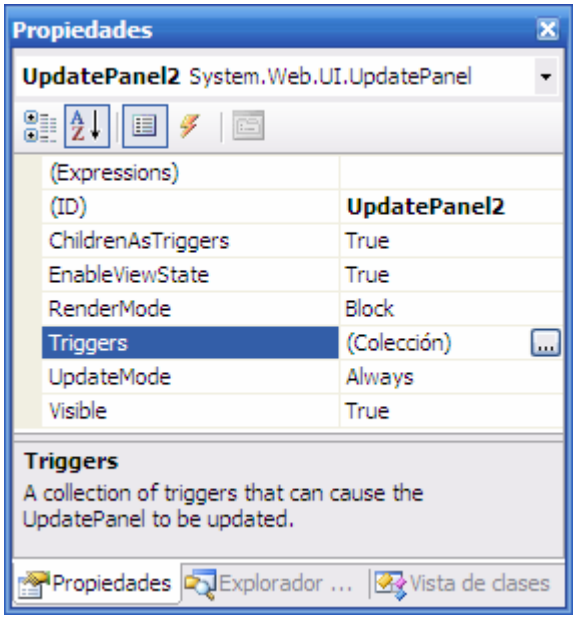


Fig. 18 UpdatePanel.

La propiedad UpdateMode puede tener el valor “Always” o el valor “Conditional”, e indica si el UpdatePanel se ha de refrescar en cada postback asíncrono o si por el contrario únicamente como resultado de una acción especifica, como por ejemplo una llamada a UpdatePanel.Update().

Por otro lado, la propiedad ChildrenAsTriggers indica si los postback provenientes de controles UpdatePanel que se encuentra dentro del control UpdatePanel en cuestión provocan un refresco/la actualización del mismo.

Y mediante la propiedad Triggers podemos especificar que eventos o modificaciones harán que se desencadene la actualización del UpdatePanel. Triggers

UpdateProgress

El control UpdateProgress detecta si se ha hecho cualquier petición fuera de banda y nos deja mostrar un mensaje de espera: en cuanto se recibe la respuesta del servidor, dicho mensaje se oculta. De esta forma, mejora la interactividad con el usuario, ya que éste es consciente de que su petición se está procesando (algo así como el icono de reloj de arena de Windows).

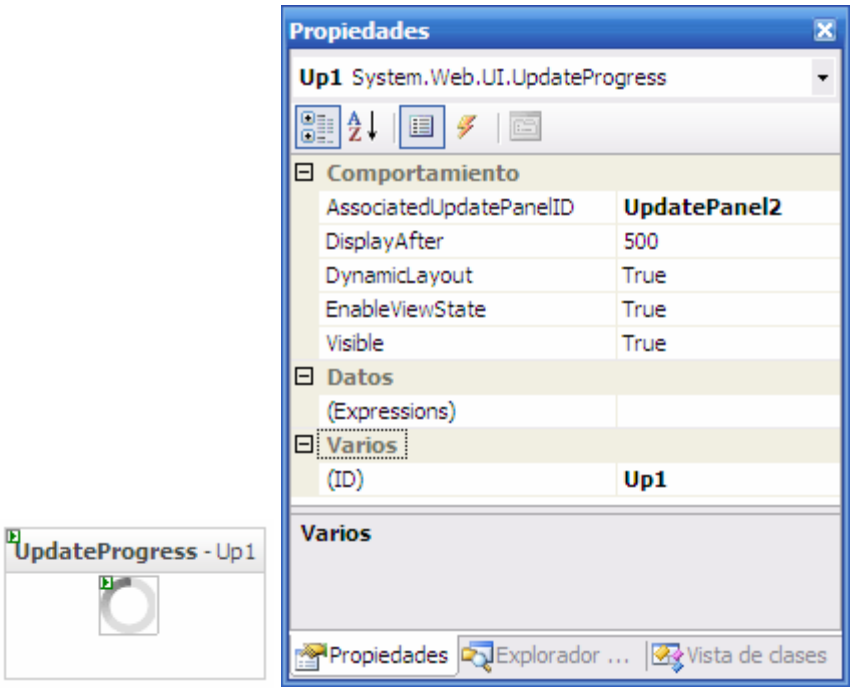


Fig. 19 UpdateProgress.

7.2.4. ASP .NET Control Toolkit

El Control Toolkit de ASP.Net AJAX es un proyecto desarrollado por la comunidad de Microsoft de código abierto. Consiste en una colección de ejemplos y componentes que facilitan el trabajar con controles y extensores que usen AJAX. Proporciona un conjunto de ejemplos listos para funcionar así como un poderoso SDK para simplificar la creación y reutilización de dichos controles y extensores. Para poder usarlo tenemos que tener primero instalado ASP.NET AJAX 1.0. Podemos descargarnos el ASP.NET Control Toolkit de sitio Web oficial de ASP.NET AJAX. (http://www.asp.net/ajax/downloads/) (*Ver Anexo 1)*

Un extender (extensor) es un control que añade funcionalidad a un control estándar ASP.NET / HTML.

A continuación se puede encontrar una descripción de las propiedades de los extensores utilizados en la aplicación.

RoundedCornersExtender

Le añade esquinas redondeadas a un panel, podemos elegir el color y el grado de curvatura de las esquinas. Hay que tener en cuenta que para lograr esto inserta elementos antes y después del panel seleccionado, con lo que la altura total del panel cambia ligeramente. (*Ver Capítulo 4, Sección Página Maestra*).

Propiedades de RoundedCornersExtender

- TargetControlID – ID del control/panel al que queremos añadir esquinas redondeadas.
- Radius – Radio de las esquinas (y altura del área añadida) .Por defecto es 5.
- Color – Color de fondo de las esquinas, por defecto coge el color de fondo del panel al que está asignado.
- Corners – Las esquinas del panel que serán redondeadas (pueden ser None, TopLeft, TopRight, BottomRight, BottomLeft, Top, Right, Bottom, Left o All).
- BorderColor – Color del borde del panel al que aplicaremos el efecto de esquinas redondeadas.

Accordion Control

El control acordeón nos permite tener varios paneles y que sólo uno sea visible a la vez. Si pinchamos en otro panel, el que está visible se esconce y se muestra el seleccionado. El control Accordion es implementado como un control Web que contiene controles AccordionPane. Cada AccordionPane tiene una plantilla para su cabecera y una para su contenido.

También lo podemos utilizar como destino de datos (DataBound), simplemente tenemos que especificar una fuente de datos en las propiedades DataSource o DataSourceID del control, y a continuación configurar los datos que se van a mostrar en las propiedades de las plantillas de la

cabecera y el contenido (HeaderTemplate y ContentTemplate). (*Ver Capítulo 4, Sección Página RSS*).

CalendarExtender

CalendarExtender puede asociarse con cualquier control TextBox. Proporciona en el lado el cliente una funcionalidad para introducir fechas con un formato personalizable de manera cómoda y sencilla. El cliente puede interactuar con el calendario pulsando un día para seleccionar esa fecha, o pulsando el link “Today” para seleccionar la fecha actual.

Además las flechas de desplazamiento permiten seleccionar un mes posterior o anterior al actual. Si pulsamos una vez en el título del calendario podemos cambiar la vista de días en el mes actual a meses en el año actual; si volvemos a pulsar cambiamos a la vista de años en décadas. Estas acciones permiten al usuario saltar de manera sencilla a fechas en el pasado o en el futuro dentro del control calendario. Con la propiedad Format del extender establecemos el formato de la fecha. (*Ver Capítulo 4, Sección Gestión de Eventos*).

MaskedEditExtender

Este extender añade funcionalidad al control TextBox. Cuando lo usamos la entrada del TextBox está condicionada a una máscara y el valor introducido es validado en el cliente de acuerdo con el tipo de dato elegido.

Para conseguir la validación, es extender está asociado con un MaskedEditValidator, que verifica la entrada y la suministra un conjunto de validaciones ASP.NET MaskedEditValidator es un control de validación asociado al extender y a la caja de texto. Verifica si los datos son coherentes en diferentes aspectos (por ejemplo, si la fecha proporcionada es una fecha valida o si una dirección de correo electrónico tiene el formato correcto). Una vez que el MaskedEditValidator está sujeto a un grupo de validación podemos efectuar una validación en el lado del servidor y en el del cliente de la entrada y mostrar los errores en el formato deseado.

Hereda de la clase BaseValidator y realiza cinco clases de validación: Required, RegularExpression, Compare o Length, DataType, y Custom. (*Ver Capítulo 4, Sección Gestión de Encuesta*).

PasswordStrengthExtender

Este extender le añade funcionalidad a un Textbox usado para la entrada de contraseñas. Podemos chequear si una contraseña cumple con los requisitos de seguridad corporativos. Además, conforme el usuario va tecleando, calcula en tiempo real el nivel de seguridad de la contraseña, los tipos de caracteres que debería de introducir y lo indica en forma de texto o de barra de progreso (*Ver Capítulo 4, Sección Controles de Sesión ASP .NET*).

Propiedades de PasswordStrengthExtender

- TargetControlID - ID del TextBox que queremos extender

- DisplayPosition – Posición del indicador de seguridad de la clave (a la izquierda / derecha del Textbox).

- StrengthIndicatorType – Tipo de indicador para medir la seguridad de la clave, una barra de progreso o texto (BarIndicator o Text).

- PreferredPasswordLength – Longitud mínima de la clave.

- PrefixText – Cabecera a añadir si tenemos un StrengthIndicatorType=Text.

- TextCssClass – Estilo a aplicar si tenemos StrengthIndicatorType=Text.

- MinimumNumericCharacters – Mínimo de caracteres numéricos que el usuario debe introducir.

- MinimumSymbolCharacters - Mínimo de caracteres “símbolo” que el usuario debe introducir (Ej. $ ^ *)

- RequiresUpperAndLowerCaseCharacters – Específica si obligamos al usuario a introducir caracteres en mayúsculas y en minúsculas.

- TextStrengthDescriptions – La seguridad de una clave se mide en un rango de valores de 2 a 10 (de menos a más). Aquí podemos indicarle un literal a cada valor, para ello le pasamos en un string una lista de descripciones separadas por ";".

- CalculationWeightings – Lista de valores numéricos separados por ";" que indican como se determina la fortaleza de una contraseña. Hay 4 características, que deben sumar 100% en total. El valor por defecto es 50;15;15;20. En ese caso damos un valor del 50% a la longitud de de la contraseña, un 15% a la cantidad de número en la contraseña, un 20% al criterio de mayúsculas/minúsculas y un 20% al criterio de símbolos. Así que el formato es 'A;B;C;D' donde A= length weighting, B = numeric weighting, C = casing weighting, D = symbol weighting.

- BarBorderCssClass – Estilo a aplicar al marco de la barra de progreso cuando StrengthIndicatorType=BarIndicator

- BarIndicatorCssClass – Estilo a aplicar a la barra de progreso cuando StrengthIndicatorType=BarIndicator

- HelpStatusLabelID – Estilo a aplicar a la etiqueta que muestra información de ayuda.

- HelpHandleCssClass – Estilo a aplicar al elemento que muestra una caja de diálogo con las reglas para dar por válida una clave.

- HelpHandlePosition – Posición del elemento/botón de ayuda (a la izquierda/derecha del TextBox).

CAPÍTULO 4: DESARROLLO DEL PORTAL DE NOTICIAS

1. ARQUITECTURA DEL PROYECTO

1.1. Introducción

La aplicación desarrollada en este proyecto fue diseñada siguiendo el modelo típico de tres capas: capa de acceso a datos, capa de lógica de negocio y capa de presentación (interfaz de usuario).

El modelo de datos a usar estará basado en tablas alojadas en una base de datos diseñada en el lenguaje SQL, utilizando para ello la herramienta SQL Server.

Dichas tablas guardarán una serie de relaciones destinadas a la simplificación y organización de toda la información requerida por las capas superiores. Para ello, se emplearán elementos como la definición de ciertos campos como Primary Key, que ayudarán a mantener una correcta indexación y coherencia de los datos.

Principalmente serán necesarias dos bases de datos: una para albergar los datos relacionados con la gestión usuarios y la administración del sitio Web; y otra para almacenar toda la información a manejar por la aplicación.

Para interactuar con la base de datos, crearemos la **capa de acceso a datos**. Estará compuesta por un conjunto de clases que serán, en su mayor parte, coincidentes con las tablas existentes en la propia base. Cada clase contendrá los métodos necesarios para intercambiar datos con la base de datos.

La **capa de lógica de negocio**s es donde se establecen y comprueban todas las reglas que deben cumplirse. Es aquí donde se manipulan los datos antes de enviarlos a la capa de presentación, siguiendo para ello una serie de criterios. Esta capa se comunica con la de datos (solicitando al gestor de la base de datos información o actualizándola) y también con la de presentación (recibiendo solicitudes o presentando resultados).

La **capa de presentación** contiene todos los elementos que configuran la interfaz del usuario. Es por tanto lo que está a la vista del cliente. Se compondrá de varias páginas Web y diferentes

controles de usuario destinados a facilitar la interacción. Esta capa se comunica exclusivamente con la capa de lógica de negocio. En la **Fig. 20** podemos observar los elementos que componen el conjunto de la arquitectura, así como la forma de interactuar entre ellos.

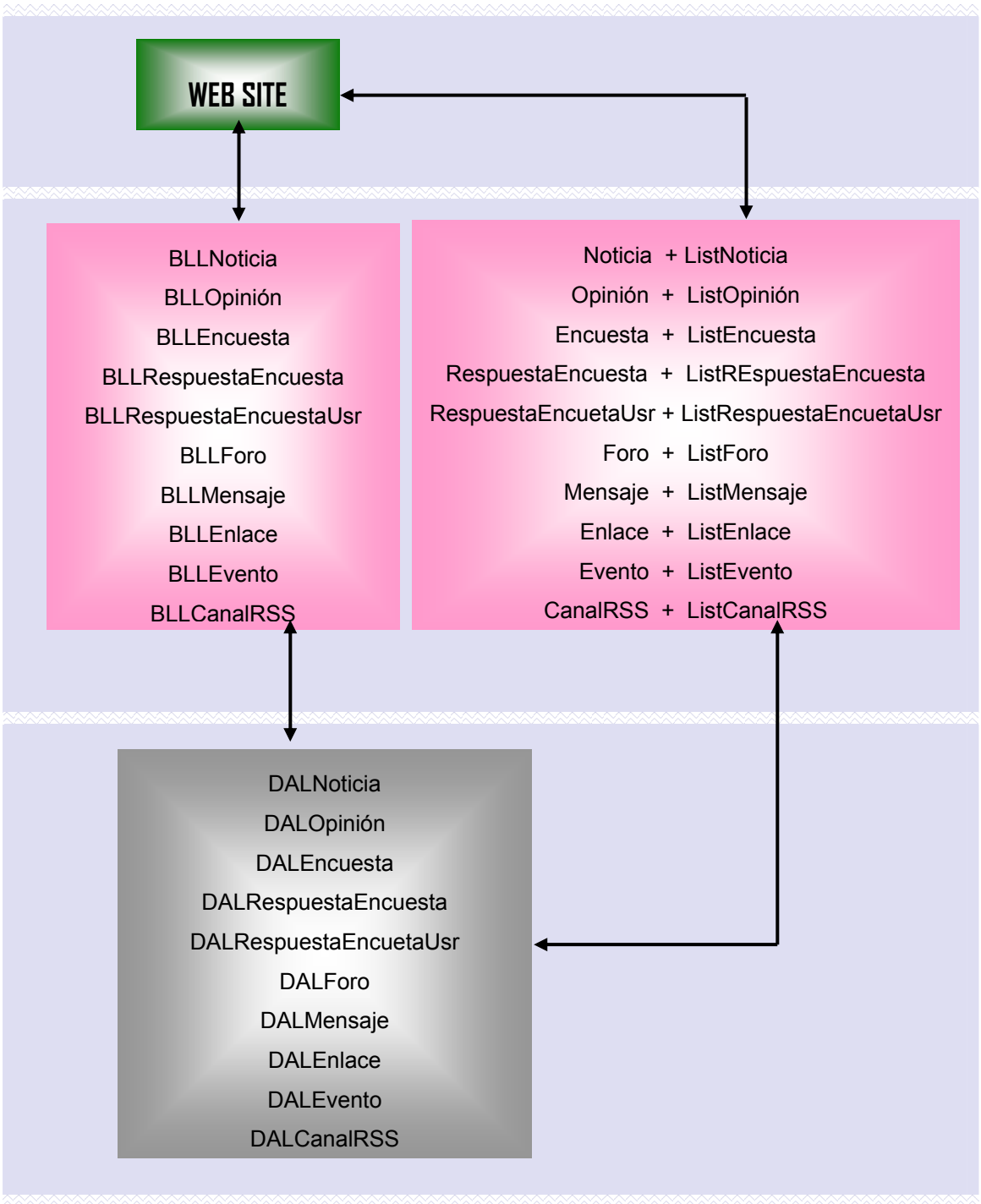


Fig. 20 Arquitectura del Sitio Web.

1.2. Requisitos

1.2.1. Requisitos Software

Para el desarrollador

- Microsoft Visual Studio 2005 Team Suite
- Plataforma: Microsoft .NET Framework. Versión 2.0.
- Lenguaje: Visual C#
- IIS (Internet Information Server)
- SQL Server
- Microsoft ASP .NET 2.0 AJAX Extension 1.0
- AJAX Control Toolkit
- Editor de texto: FCKeditor .Net_2.5
- Administración de Miembros y Funciones: MembershipEditor
- Navegador: Internet Explore Versión 5 o superior, Mozilla, etc.

Para el usuario

- Navegador: Internet Explore Versión 5 o superior, Mozilla, etc.

1.2.2. Requisitos de implementación

- Arquitectura típica en tres capas: capa de acceso de datos (en contacto con la base de datos), capa de negocios, y capa de presentación (las páginas y controles de usuario).

- Administración mediante diferentes roles ajustados a cada de tipo de usuario para los que asignará diferentes permisos y accesos (administrador, moderador, y usuarios).

- Debe ser capaz de recoger y mostrar noticias obtenidas de diferentes fuentes RSS, así como ofrecer la posibilidad de suscribirse a nuestro propio canal de RSS.

- Inclusión de controles de autenticación y validación de usuarios basados en formularios.

- Posibilidad por parte de los usuarios de personalizar su interfaz del portal.

- Un control de búsqueda de contenidos.

- Posibilitar la participación por parte de los usuarios en los contenidos de la Web mediante diferentes mecanismos:
    - Un foro de opinión con diversas secciones.
    - Comentarios sobre noticias o elementos concretos.
    - Votaciones sobre encuestas.

Además de todos los requerimientos antes mencionados, para el desarrollo del Portal se contó con lo siguientes recursos:

- Computadora HP Pavilion dv200:
    - Disco Duro: 120 GB.
    - Memoria RAM: 1 GB
    - Procesador: Intel® Core™ 2. 1.66 GHz.
- Sistema Operativo: Microsoft Windows XP. Media Center Edition. Versión 2.0.
- Service Pack 2.
- Microsoft Office Profesional Edition 2003.
- Adobe Acrobat 6.0 Profesional.
- Windows Internet Explorer 7.

2. MODELO DE DATOS

2.1. Introducción

Cuando se va ha desarrollar una aplicación Web, la cual será capaz de almacenar datos, se debe poner mucho énfasis en el diseño del modelo de datos, pues sobre él recaerá todo lo concerniente a la manipulación de la información de nuestra aplicación, y no menos importante es la tarea de elegir el sistema de gestión de base de datos.

Se utilizará Microsoft SQL Server 2005 como gestor de base de datos, pues es muy potente, pero sencillo de utilizar y además utiliza el lenguaje SQL, que con el cual el estudiante ya está familiarizado.

El diseño propuesto es el resultado de un análisis previo, sobre los requerimientos del portal y de los conocimientos que debe adquirir el estudiante en este componente curricular.

El proyecto se compone de dos base de datos: la primera (el fichero *BD_DATOS.mdf*) responsable de almacenar todo el contenidos del portal y que el usuario podrá tener acceso a ella mediante la capa de presentación.

La segunda (el fichero *ASPNET.MDF*), que almacena toda la información relacionada con la gestión de usuarios y roles que proporciona ASP.NET 2.0, la cual es creada por el Asistente de Configuración de ASP .NET desde el Visual Studio y de ella dependerán todos los controles de usuario.

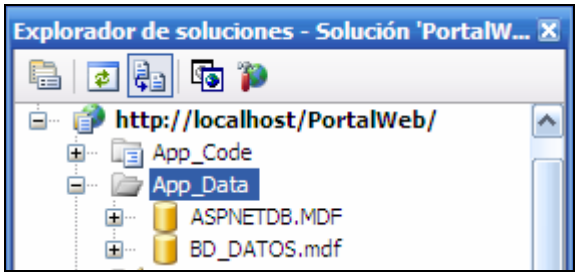


Fig. 21 Bases de datos del Portal.

2.2. Base de datos BD_DATOS.mdf

Para agregar una base de datos a nuestro Sitio Web (*Ver Anexo 1*), clic derecho sobre la carpeta *App_Data* → *Agregar nuevo elemento*…, en el cuadro de diálogo seleccione como Plantilla **Base de datos SQL**, a continuación asígnele un nombre (*BD_DATOS*) de clic en **Agregar**

La **Fig. 23** muestra el diagrama de diseño de la base de datos BD_DATOS.mdf, en el que se puede observar las diferentes tablas que constituyen la base de datos, así como también los campo de cada uno de las tablas y las relaciones entre las misma. Para crear el diagrama de diseño, estando en la vista *Explorador de Servidores*, expanda el nodo correspondiente a la Base de Datos creada anteriormente, luego sobre el nodo *Diagrama de base de datos*, de clic derecho y seleccione *Agregar nuevo diagrama,* en el cuadro de diálogo que se nos muestra, deberá seleccionar las tablas que formarán parte del diagrama dar clic en **Agregar** y finalmente en **Cerrar.**

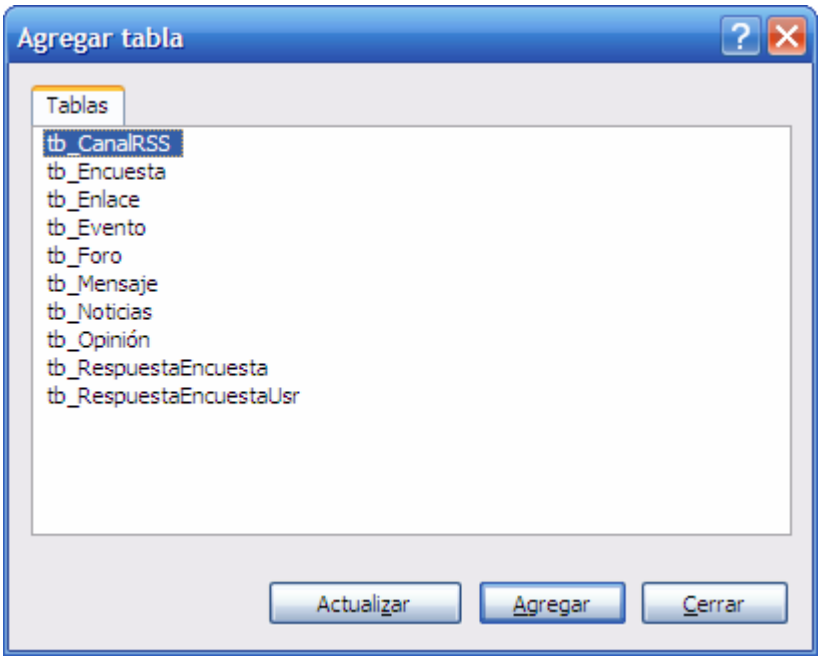


Fig. 22 Crear un Diagrama de Base de Datos

Cuando termine con la edición, guarde los cambios y asígnele un nombre (*Diagrama_BD_DATOS*). Recuerde que este diagrama se realizará al finalizar la creación de todas las tablas que componen la base de datos.

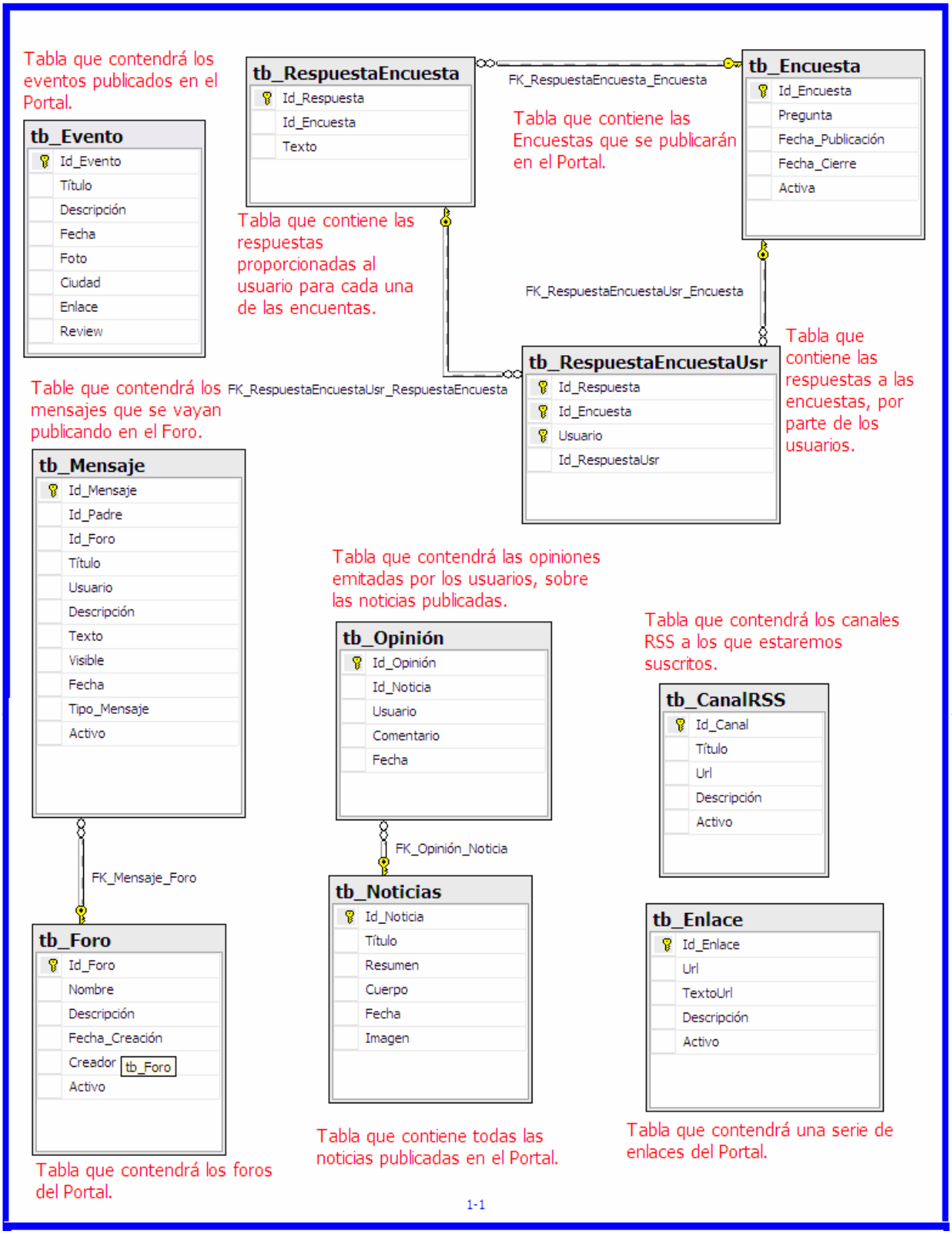


Fig. 23 Diagrama de Base de Datos.

2.2.1. Definición de las tablas

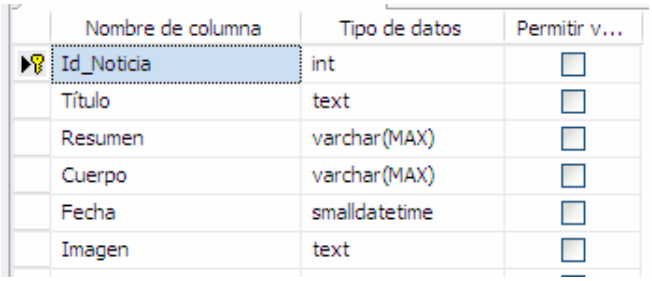
Estando en la vista *Explorador de Servidores*, para crear una tabla, realice los siguientes pasos:

- Clic derecho sobre el nodo *Tablas*
- *Agregar nueva tabla*
- Defina el Nombre de la columna, Tipo de datos, Permitir valores nulos.
- Guarde los cambios asignándole un nombre a la nueva tabla.

A continuación se describirán cada una de las tablas que componen la base de datos, siga los pasos descritos anteriormente para crearlas, tome en cuenta el tipo de datos, las claves primarias y la especificación de permitir o no valores nulos.

Tabla tb_Noticias

Almacenará toda la información correspondiente a las noticias que se publicarán en el Portal.



Nombre de columna	Tipo de datos	Permitir v...
Id_Noticia	int	☐
Título	text	☐
Resumen	varchar(MAX)	☐
Cuerpo	varchar(MAX)	☐
Fecha	smalldatetime	☐
Imagen	text	☐

Fig. 24 Definición tabla tb_Noticias.

- **Id_Noticia:** Identificador único de cada noticia., declarado tipo *int* y *autonumérico*. Será la clave principal de la tabla.
- **Título:** Albergará el título de la noticia.
- **Resumen**: Pequeño resumen de la noticia.
- **Cuerpo**: Cuerpo de la noticia. Tanto este como el campos anterior, debido al posible gran tamaño del texto a almacenar han sido declarados de tipo *varchar(MAX)*.
- **Fecha**: Fecha de publicación de la noticia.
- **Imagen**: Path (ruta) de la imagen asociada a la noticia.

Todo los campos son obligatorios para cada registro de la tabla por lo que no permiten valores nulos.

Para especificar que un campo o columna funciona como clave primaria, seleccione dicha columna y de clic sobre el icono *Establecer clave primaria*  .

Para declarar un tipo autonumérico, debe modificar la propiedad identidad del campo deseado. Cuando se insertan valores en una tabla que contiene una columna de este tipo, Microsoft SQL Server genera automáticamente el siguiente identificador a partir de los últimos valores de identidad (la propiedad Inicialización de identidad) y de incremento (la propiedad Incremento de identidad) especificados en las propiedades de la columna.

Para modificar las propiedades de identidad de una columna:

a. En la ficha **Propiedades de columna**, expanda la propiedad **Especificación de identidad**.
b. Haga clic en la celda de la cuadrícula correspondiente a la propiedad secundaria **Identidad** y elija **Sí** en la lista desplegable.
c. Escriba un valor en la celda **Inicialización de identidad**. Este valor se asignará a la primera fila de la tabla. De manera predeterminada, se asignará el valor 1.
d. Escriba un valor en la celda **Incremento de identidad**. Este valor es el incremento que se agregará a **Inicialización de identidad** en cada fila siguiente. De manera predeterminada, se asignará el valor 1.

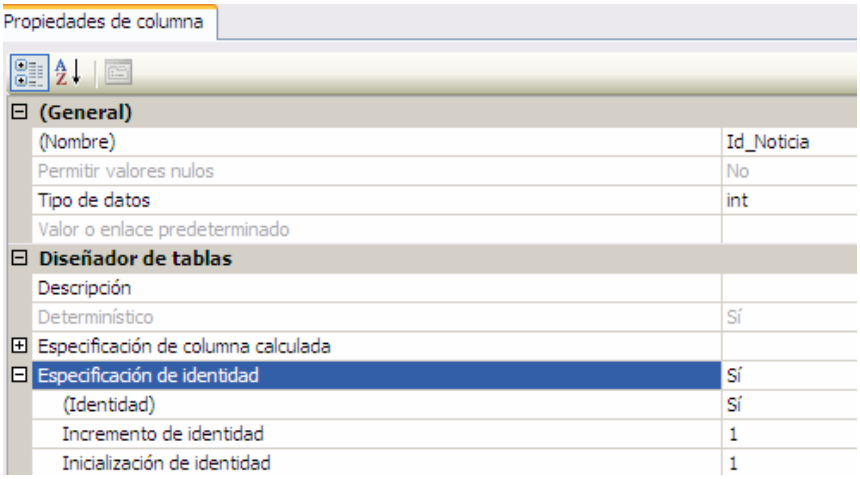
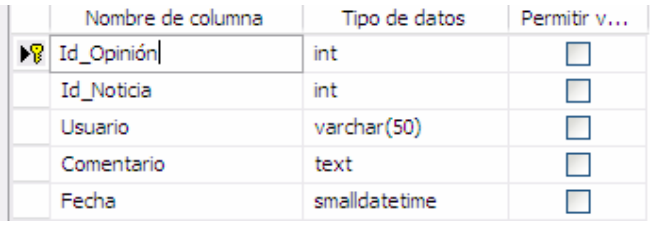


Fig. 25 Especificación de identidad.

Tabla tb_Opinión.

Para crear y establecer las propiedades de las tablas descritas a continuación, siga los pasos realizados para la tabla tb_Noticias. La tabla tb_Opinión, contendrá cada una de las opiniones emitidas por los usuarios sobre las noticias publicadas en el Portal.



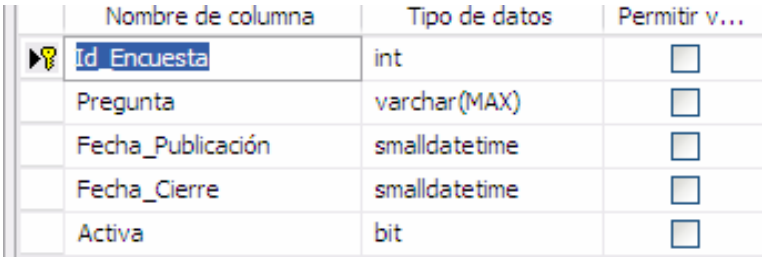
Nombre de columna	Tipo de datos	Permitir v...
Id_Opinión	int	☐
Id_Noticia	int	☐
Usuario	varchar(50)	☐
Comentario	text	☐
Fecha	smalldatetime	☐

Fig. 26 Definición tabla tb_Opinión.

- **Id_Opinión:** Identificador único de cada una de las opiniones, declarado de tipo int y autonumérico, constituye la clave principal de la tabla.
- **Id_Noticia:** Identificador de la noticia asociada a cada opinión. Con este campo y su homólogo en la tabla tb_Noticias, se asociarán ambas tablas.
- **Usuario:** Nombre del usuario que emite la opinión.
- **Comentario:** Texto que contiene el comentario introducido por el usuario.
- **Fecha:** Fecha de inserción de la opinión. Es del tipo *smalldatetime* para tener una mayor precisión (almacena hasta los segundos) a la hora de ordenar las opiniones cronológicamente.

Tabla tb_Encuesta.

Tabla dirigida al almacenamiento de las encuestas que se publicarán en el Portal.



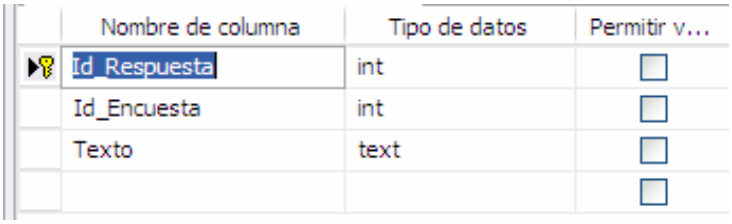
Nombre de columna	Tipo de datos	Permitir v...
Id_Encuesta	int	☐
Pregunta	varchar(MAX)	☐
Fecha_Publicación	smalldatetime	☐
Fecha_Cierre	smalldatetime	☐
Activa	bit	☐

Fig. 27 Definición tabla tb_Encuesta.

- **Id_Encuesta:** Identificador único de cada una de las encuestas publicadas en el Portal, es un dato de tipo int y autonumérico. Representa la clave principal de la tabla.
- **Pregunta:** Pregunta o título de la encuesta.
- **Fecha_Publicación:** Fecha en la cual se publica la encuesta.
- **Fecha_Cierre:** Fecha en la cual no se permitirán más participaciones en la encuesta.
- **Activa:** Campo de tipo bit, que determina si la encuesta está o no activa.

Tabla tb_RespuestaEncuesta.

Almacenará las posibles respuestas para cada una de las encuestas.



Nombre de columna	Tipo de datos	Permitir v...
Id_Respuesta	int	☐
Id_Encuesta	int	☐
Texto	text	☐
		☐

Fig. 28 Definición tabla tb_RespuestaEncuesta.

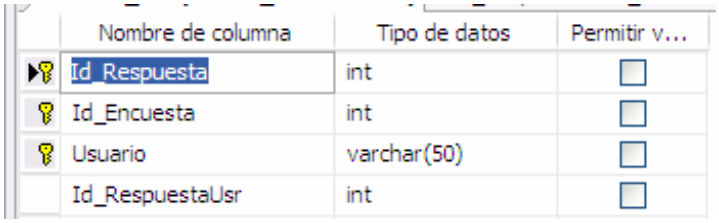
- **Id_Respuesta:** Identificador único de cada respuesta asociada a las encuestas, constituye la clave principal de la tabla el cual deberá ser autonumérico.
- **Id_Encuesta:** Identificador de la encuesta asociada a cada una de las respuestas, mediante este campo y su homólogo en la tabla tb_Encuesta, se relacionarán ambas tablas.
- **Texto:** Contendrá el texto de la posible respuesta.

Tabla tb_RespuestaEncuestaUsr

Almacenará las respuestas seleccionadas por los usuarios, asociadas a las diferentes encuestas.

- **Id_Respuesta:** Identificador correspondiente a la Respuesta seleccionada por el usuario, con este campo y el correspondiente en la tabla tb_RespuestaEncuesta, se relacionarán ambas tablas.

- **Id_Encuesta:** Identificador correspondiente a la encuesta, sobre la cual ha votado el usuario. Con este campo y su homólogo en la tabla tb_Encuesta se relacionarán ambas tablas.



Nombre de columna	Tipo de datos	Permitir v...
Id_Respuesta	int	
Id_Encuesta	int	
Usuario	varchar(50)	
Id_RespuestaUsr	int	

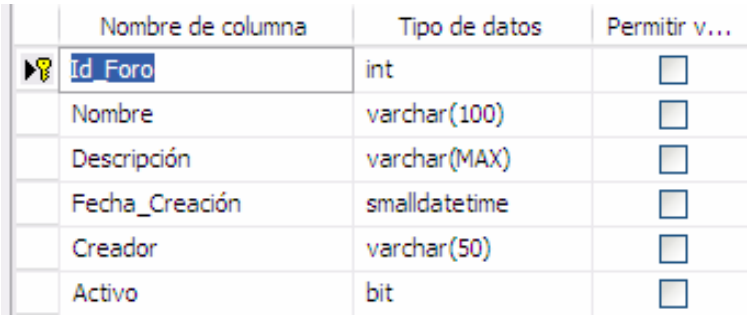
Fig. 29 Definición tabla tb_RespustaEncuestaUsr.

- **Usuario:** Nombre del usuario que participa en la encuesta.
- **Id_RespuestaUsr:** Identificador único de cada registro de la tabla. Lo declaramos tipo entero int y autonumérico.

Los campos Id_Respuesta, Id_Encuesta, y Usuario están definidos como clave principal. De esta manera aseguramos que no se repita la misma combinación de valores para estos campos en más de un registro.

Tabla tb_Foro

Contendrá cada uno de los foros existentes en la sección Foros.



Nombre de columna	Tipo de datos	Permitir v...
Id_Foro	int	
Nombre	varchar(100)	
Descripción	varchar(MAX)	
Fecha_Creación	smalldatetime	
Creador	varchar(50)	
Activo	bit	

Fig. 30 Definición tabla tb_Foro.

- **Id_Foro:** Identificador único de cada uno de los foros publicados en el Portal. Constituye la clave principal de la tabla, debe ser autonumérico.
- **Nombre:** Nombre asociado al foro.

- **Descripción:** Pequeña descripción sobre los temas a tratar en el foro.
- **Fecha_Creación:** Fecha en la que se crea el foro.
- **Creador:** Nombre del usuario que creará el foro.
- **Activo:** Valor de tipo bit (Verdadero/Falso) que nos indicará si el foro continúa o no activo.

Tabla tb_Mensaje

Tabla destinada a almacenar y recoger cada uno de los detalles correspondientes a los mensajes que se publiquen en los foros.

Nombre de columna	Tipo de datos	Permitir v...
Id_Mensaje	int	☐
Id_Padre	int	☑
Id_Foro	int	☑
Título	varchar(MAX)	☑
Usuario	varchar(MAX)	☑
Descripción	varchar(MAX)	☑
Texto	text	☑
Visible	bit	☑
Fecha	smalldatetime	☑
Tipo_Mensaje	int	☑
Activo	bit	☑
		☐

Fig. 31 Definición tabla tb_Mensaje.

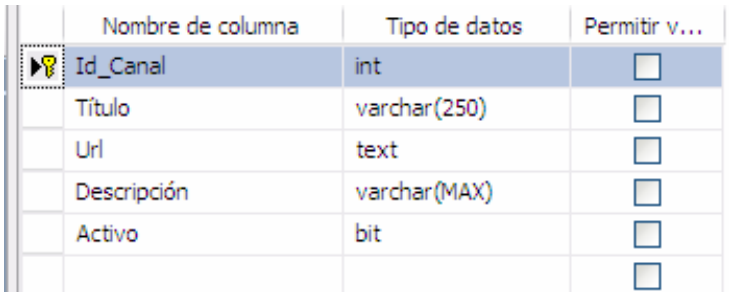
- **Id_Mensaje:** Identificador único de cada mensaje, constituye la clave principal de la tabla, de tipo int y autonumérico.
- **Id_Padre:** Valor correspondiente al número asignado al mensaje padre del que puede o no depender un mensaje, dependiendo de si es subyacente de uno ya existente o si es nuevo.
- **Id_Foro:** Identificador del foro en el cual se encuentra ubicado el mensaje. Constituye la vía de relación entre la tabla tb_Foro y tb_Mensaje.
- **Título:** Título con el que figurará el mensaje, sólo tendrá cabida si se trata de un nuevo mensaje.
- **Usuario:** Nombre del usuario que emite el mensaje.

- **Descripción:** Breve descripción del mensaje, sólo tiene sentido cuando se trata de un nuevo mensaje.
- **Texto:** Cuerpo del mensaje.
- **Visible:** Valor de tipo Verdadero/Falso que nos indica si dicho mensaje puede ser visto por todos los usuarios del foro o si por el contrario es un mensaje oculto.
- **Fecha:** Fecha en la que se emitió el mensaje.
- **Tipo_Mensaje:** no servirá para diferenciar si se trata de un mensaje normal, de un mensaje de relevancia, de un anuncio, etc. Cada tipo corresponderá con un entero.
- **Activo:** Nos indicará si el mensaje sigue activo o no dentro del ciclo de vida del foro.

Todos lo campos de esta tabla pueden contener valores nulos, excepto la clave primaria.

Tabla tb_CanalRSS.

Almacenará los nombres de los canales RSS, a los que estaremos suscritos desde el Portal.



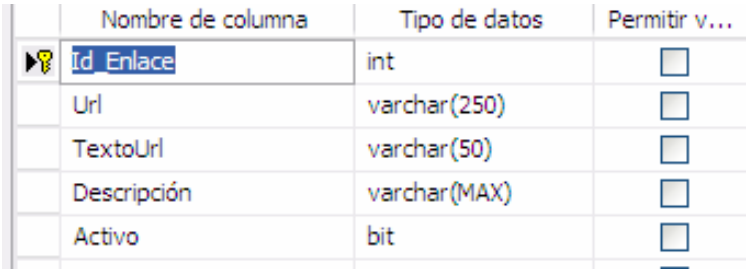
Nombre de columna	Tipo de datos	Permitir v...
Id_Canal	int	☐
Título	varchar(250)	☐
Url	text	☐
Descripción	varchar(MAX)	☐
Activo	bit	☐
		☐

Fig. 32 Definición tb_CanalRSS.

- **Id_Canal:** Identificador único de canal RSS en la tabla, constituye la clave primaria de la tabla, deberá ser autonumérico.
- **Título:** Título del canal RSS.
- **Url:** Url correspondiente a la fuente del canal.
- **Descripción:** Breve descripción del contenido del canal.
- **Activo:** Indica si el canal está o no activo.

Tabla tb_Enlace

Almacenará la información sobre distintos enlaces Web, a los que podremos acceder desde el Portal.



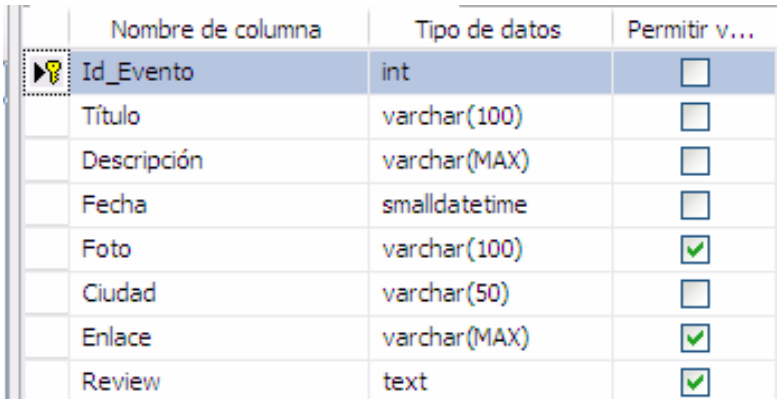
Nombre de columna	Tipo de datos	Permitir v...
Id_Enlace	int	☐
Url	varchar(250)	☐
TextoUrl	varchar(50)	☐
Descripción	varchar(MAX)	☐
Activo	bit	☐

Fig. 33 Definición tabla tb_Enlace.

- **Id_Enlace:** Identificador único de cada enlace en el Portal. Constituye la clave primaria de la tabla, de tipo int y autonumérico.
- **Url:** Url correspondiente al enlace.
- **TextoUrl:** Texto que aparecerá como título del enlace.
- **Descripción:** Breve descripción sobre el enlace.
- **Activo:** Indica si el enlace está o no activo.

Tabla tb_Evento

Almacenará la información de los eventos publicado en el Portal.



Nombre de columna	Tipo de datos	Permitir v...
Id_Evento	int	☐
Título	varchar(100)	☐
Descripción	varchar(MAX)	☐
Fecha	smalldatetime	☐
Foto	varchar(100)	☑
Ciudad	varchar(50)	☐
Enlace	varchar(MAX)	☑
Review	text	☑

Fig. 34 Definición tabla tb_Evento.

- **Id_Evento:** Identificador único de cada evento, constituye la clave primaria de la tabla, de tipo int y autonumérico.
- **Título:** Título del evento publicado.
- **Descripción:** Una pequeña descripción de lo que será el evento.
- **Fecha:** Fecha en la cual se llevará a cabo el evento.
- **Foto:** Foto relacionada con el evento.
- **Ciudad:** Ciudad en la cual se llevará a cabo el evento.
- **Enlace:** Enlace (URL) a la Web oficial del evento.
- **Review:** Pequeña anécdota sobre los eventos que se han llevado a cabo.

2.2.2. Definición de relaciones entre las tabla

Una vez definidas las tablas, realice el Diagrama de base de datos, según se indicó anteriormente, este será el punto de partida para la creación de las relaciones entre las tablas de la base de datos.

Estando en la vista *Explorador de Servidores*, para crear una relación, realice los siguientes pasos:

- Expanda el nodo *Diagramas de base de datos*.
- Doble clic sobre el diagrama (*Diagrama_BD_DATOS*).
- Seleccione en la tabla que contiene la clave primaria, este campo y arrástrelo hasta la tabla que contiene la llave externa.
- Asígnele un nombre a la relación.
- Comprueba que la tabla principal y la externa sean correcta.
- Compruebe que el campo por el cual está relacionando ambas tablas sea el correcto.
- Luego clic en aceptar.

Relación FK_Noticia_Opinión

Las tablas tb_Noticias y tb_Opinión están relacionadas mediante la relación FK_Noticia_Opinión. Como vemos en la **Fig. 35** , la relación es del tipo uno a varios. Relacionamos un registro de la tabla tb_Noticias con varios registros de la tabla tb_Opinión.

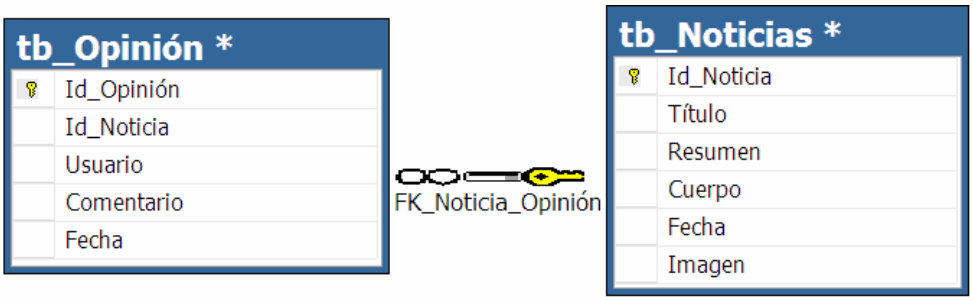


Fig. 35 Relación FK_Noticia_Opinión.

La relación se establece entre el campo Id_Noticia de la tabla tb_Noticias y el campo Id_Noticia de la tabla tb_Opinión.

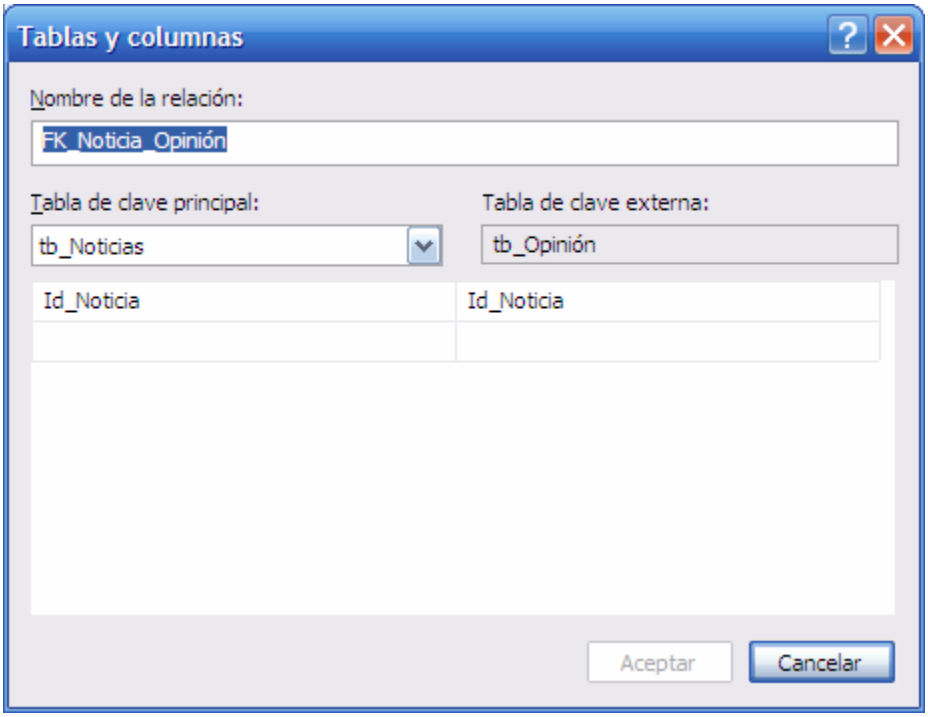


Fig. 36 Creación de la relación entre las tablas tb_Noticias y tb_Opinión.

Relación FK_Encuesta_RespuestaEncuesta

Las tablas tb_Encuestas y tb_RespuestasEncuesta están relacionadas mediante la relación FK_ Encuesta_RespuestaEncuesta. Como vemos en la **Fig. 37**, la relación es del tipo uno a varios. Relacionamos un registro de la tabla tb_Encuesta con varios registros de la tabla tb_RespuestasEncuesta.

Fig. 37 Relación FK_Encuesta_RespuestaEncuesta.

La relación se estable mediante el campo Id_Encuesta de ambas tablas.

Relación FK_Encuesta_RespuestaEncuestaUsr

Las tablas tb_Encuesta y tb_RespuestaEncuestaUsr están relacionadas mediante la relación FK_ Encuesta_RespuestaEncuestaUsr. Como vemos en la **Fig. 38**, la relación es del tipo uno a varios. Relacionamos un registro de la tabla tb_Encuesta con varios registros de la tabla tb_RespuestaEncuestaUsr.

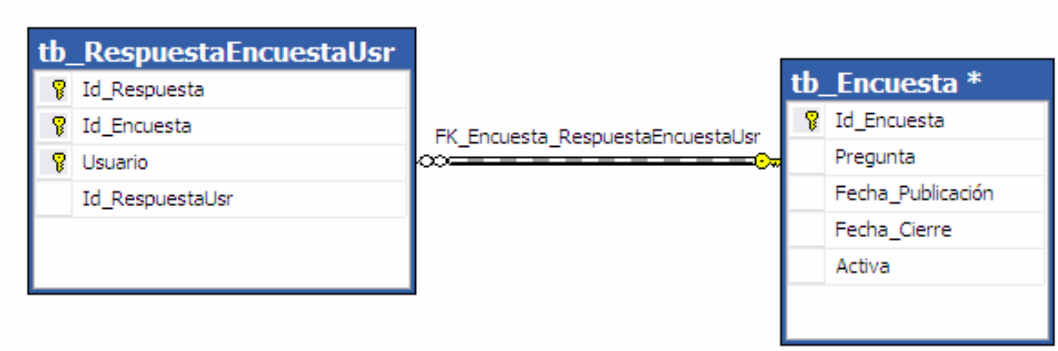


Fig. 38 Relación FK_Encuesta_RespuestaEncuestaUsr.

La relación se estable mediante el campo Id_Encuesta de ambas tablas.

Relación FK_RespuestaEncuesta_RespuestaEncuestaUsr

Las tablas tb_RespuestaEncuesta y tb_RespuestaEncuestaUsr están relacionadas mediante la relación FK_ RespuestaEncuesta_RespuestaEncuestaUsr. Como vemos en la **Fig. 39**, la

relación es del tipo uno a varios. Relacionamos un registro de la tabla tb_RespuestaEncuesta con varios registros de la tabla tb_RespuestaEncuestaUsr.

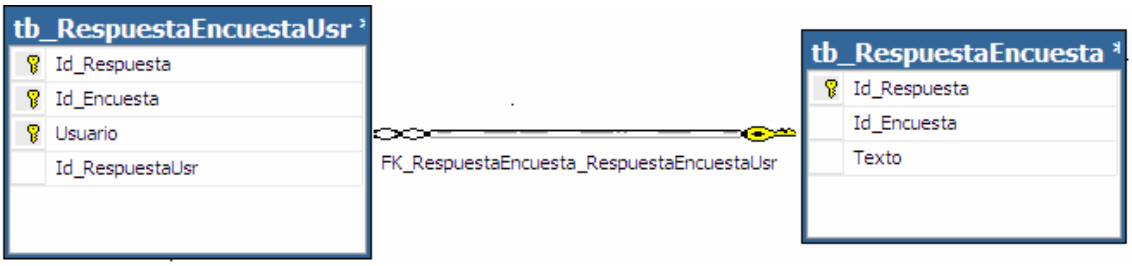


Fig. 39 Relación FK_RespuestaEncuesta_RespuestaEncuestaUsr.

La relación se estable mediante el campo Id_Repuesta de ambas tablas.

Relación FK_Foro_Mensaje

Las tablas tb_Foro y tb_Mensaje están relacionadas mediante la relación FK_Foro_Mensaje. Como se observa en la **Fig. 40**, la relación es del tipo uno a varios. Relacionamos un registro de la tabla tb_Foro con varios registros de la tabla tb_Mensaje.

La relación se establece entre el campo Id_Foro de la tabla tb_Foro y el campo Id_Foro de la tabla tb_Mensaje.

En este punto ya tiene construida la base de datos y sus correspondientes relaciones, personalice el Diagrama, agregándolo por ejemplo: *Notaciones de texto, etiquetas de las relaciones, saltos de página*, etc. (Ver **Fig. 23**).

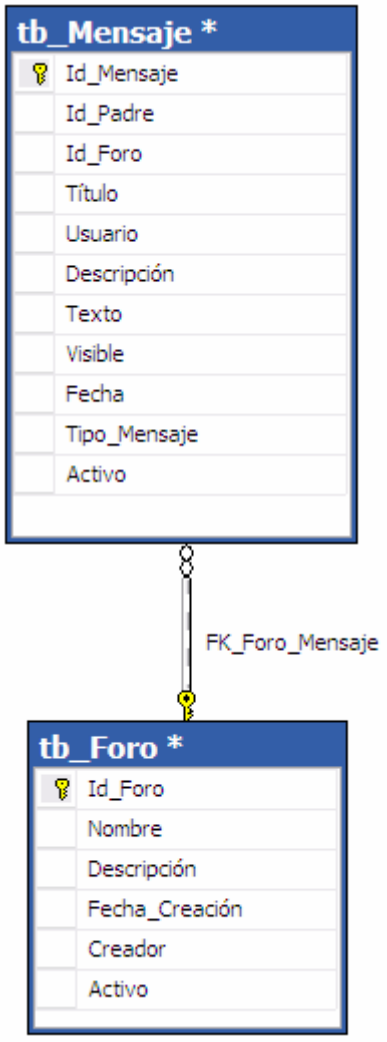


Fig. 40 Relación FK_Foro_Mensaje.

2.2.3. Procedimientos almacenados

Estando en la vista *Explorador de Servidores*, para crear un procedimiento almacenado, realice los siguientes pasos:

- Clic derecho sobre el nodo *Procedimientos almacenados.*
- Clic en *Agregar nuevo procedimiento almacenado*.
- Asígnele un nombre al procedimiento. (por ejemplo: dbo.StoredProcedure1 → dbo.sprocNoticiasDelete).
- Defina el cuerpo del procedimiento almacenado.
- Guarde los cambios realizados.

Procedimientos de la tabla tb_Noticias.

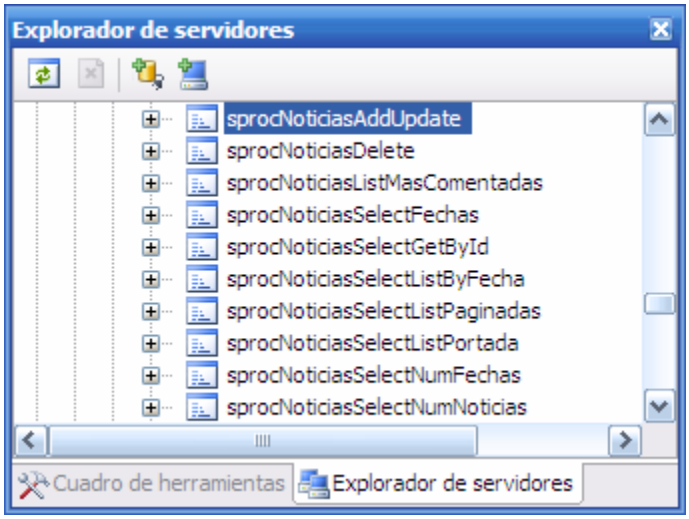


Fig. 41 Procedimientos de la tabla tb_Noticias

sprocNoticiasAddUpdate

Este procedimiento tiene la capacidad de realizar la inserción de un nuevo registro en la tabla tb_Noticias o la modificación de un existente, en dependencia del valor del parámetro id_noticia.

Recibe como parámetros los valores de los diferentes campos de la tabla, si el valor del parámetro id_noticia es NULL, realiza la inserción de un nuevo registro con los valores pasados

como parámetros, en la tabla tb_Noticias. Recuerde que el campo Id_Noticia es autonumérico, y por tanto no es precise asignarle un valor, cuando se insertará una nueva fila en la tabla.

Si el valor del parámetro id_noticia no es NULL, realiza una actualización del registro cuyo campo Id_Noticia coincida con el pasado como parámetro, los campos de este registro son actualizados con los correspondiente parámetros.

```
ALTER PROCEDURE dbo.sprocNoticiasAddUpdate

        @id_noticia int,
        @titulo text,
        @resumen varchar(max),
        @cuerpo varchar(max),
        @fecha datetime,
        @imagen text

AS
        DECLARE @ReturnValue int

        IF(@id_noticia IS NULL)
        BEGIN
        INSERT INTO tb_Noticias
         (
                Título,
                Resumen,
                Cuerpo,
                Fecha,
                Imagen
        )
        VALUES
        (
                @titulo,
                @resumen,
                @cuerpo,
                @fecha,
                @imagen
        )
        SELECT @ReturnValue = SCOPE_IDENTITY ()
        END

        ELSE
                BEGIN
                UPDATE tb_Noticias
                SET
                        Título = @titulo,
                        Resumen = @resumen,
                        Cuerpo = @cuerpo,
                        Fecha = @fecha,
                        Imagen=@imagen
                WHERE(@id_noticia = Id_Noticia)
                SELECT @ReturnValue = @id_noticia
```

```
END
        IF (@@ERROR != 0)
BEGIN
        RETURN -1
END
ELSE
BEGIN
        RETURN @ReturnValue
END
RETURN
```

sprocNoticiasDelete.

Permite eliminar una noticia del portal mediante su Id. Este procedimiento eliminará la noticia de la tabla tb_Noticias, cuyo Id coincida con el pasado como parámetro, también eliminará todas las opiniones de la tabla tb_Opinión, asociadas a la noticia que se ha eliminado.

```
ALTER PROCEDUREE dbo.sprocNoticiasDelete

        @id_noticia int

        AS
                DELETE FROM
                tb_Noticias
                WHERE(@id_noticia = Id_Noticia)

                DELETE FROM
                tb_Opinión
                WHERE (@id_noticia = Id_Noticia)
```

sprocNoticiasLisMasComentadas.

Procedimiento utilizado para obtener el Id de las noticias más comentadas, es decir aquellos Id's de noticias que más se repiten en la tabla tb_Opinión. Realiza una consulta sobre la tabla tb_Opinión, agrupando los registros de dicha tabla por el campo Id_Noticia, luego cuenta las ocurrencias que hay en cada grupo, ordenándolos de forma descendente y finalmente selecciona los 5 primeros registros.

```
ALTER PROCEDURE dbo.sprocNoticiasListMasComentadas

        AS
                SELECT     TOP (5) Id_Noticia, COUNT(1) AS Expr1
                FROM  tb_Opinion
                GROUP BY Id_Noticia
                ORDER BY Expr1 DESC
```

sprocNoticiasSelectFechas.

Procedimiento utilizado para conocer las fechas en las cuales se han publicado noticias en el Portal.

Realiza una consulta sobre la tabla tb_Noticias seleccionando el campo Fecha. Al incluir la cláusula DISCTINT, se ignorarán aquellas filas cuyo valor sea igual que el de otras. Las filas generadas de esta consulta estarán ordenadas de forma ascendente.

```
ALTER PROCEDURE dbo.sprocNoticiasSelectFechas

   AS
      SELECT DISTINCT CAST(LEFT(Fecha, 11) AS DateTime) AS Expr1
      FROM        tb_Noticias
     ORDER BY Expr1
```

sprocNoticiasSelectGetById

Procedimiento para obtener una determinada noticia de la base de datos a partir de su identificador. Realiza una consulta sobre la tabla tb_Noticias, para obtener todos los campos del registro cuyo Id_Noticia coincida con el entero pasado como argumento.

```
ALTER PROCEDURE dbo.sprocNoticiasSelectGetById
        @id_noticia int

        AS
                SELECT * FROM
                tb_Noticias
                WHERE (@id_noticia = Id_Noticia
```

sprocNoticiasSelectListByFecha

Utilizado para obtener todas las noticias publicadas en una determinada fecha. Este procedimiento realiza una consulta sobre la tabla tb_Noticias seleccionando todos aquellos registros cuyo campo Fecha se encuentre entre el rango pasado parámetro. El conjunto de resultados se ordena por el campo Id_Noticia de manera descendente.

```
sprocNoticiasSelectListByFecha
@fecha1 DateTime,
@fecha2 DateTime
```

```
AS

        SELECT    *
        FROM      tb_Noticias
        WHERE   (Fecha >= @fecha1) AND (Fecha <= @fecha2)
        ORDER BY Id_Noticia DESC
```

sprocNoticiasSelectListPortada

Procedimiento utilizado para obtener las noticias más recientes, las que se colocarán en la portada del Portal.

Realiza una consulta sobre la tabla tb_Noticias seleccionando los cinco primeros registros que se ajusten al rango especificado por la cláusula ORDER BY, es decir, los cinco registros más recientes ordenados a su vez por el Id_Noticia de forma descendente. Esta doble condición sirve para escoger los registros que habiendo sido introducidos el mismo día sean a su vez los últimos añadidos a la tabla.

```
ALTER PROCEDURE dbo.sprocNoticiasSelectListPortada
AS
        SELECT    TOP (5) *
        FROM      tb_Noticias
       ORDER BY Fecha DESC, Id_Noticia DESC
```

sprocSelectNoticiasNumFechas

Procedimiento mediante el cual conoceremos el número de fechas en las que se han publicado las noticias del portal.

Realiza una consulta sobre la tabla tb_Noticias, la cláusula Count cuenta las ocurrencias de las fechas únicas (DISTINCT), el resultado lo muestra en un campo calculado llamado Total.

```
ALTER PROCEDURE dbo.sprocNoticiasSelectNumFechas
AS
        SELECT
        COUNT(DISTINCT CAST(LEFT(Fecha, 11) AS DateTime)) AS Total
        FROM tb_Noticias
```

sprocNoticiasSelectNumNoticias

Procedimiento mediante el cual conoceremos el número de noticias publicadas en el portal. Y retorna este valor en un nuevo campo denominado numnoticias.

```
ALTER PROCEDURE dbo.sprocNoticiasSelectNumNoticias

 AS
        SELECT    COUNT(1) AS numnoticias
        FROM       tb_Noticias
```

sprocNoticiasListByIdNoticiaPaginadas

Este procedimiento almacenado, permite obtener noticias por páginas. El número de filas que se desean obtener está determinado por el parámetro maximunRows y el inicio de la selección está dado por el parámetro startRowIndex.

```
ALTER PROCEDURE dbo.sprocNoticiasSelectListPaginadas

@startRowIndex int,
@maximumRows int

AS
BEGIN
SET NOCOUNT ON;
        --Hacemos un selec sobre un rango de valores
        SELECT    *, RowRank
        FROM
        --En el from estamos diciendo que nos haga una consulta sobre la tabla tb_Noticia
        --y que nos asigne un valor de rango a cada fila del resultado
        (
        SELECT    *, (ROW_NUMBER() OVER (ORDER BY Fecha DESC)) AS RowRank
        FROM     tb_Noticias

        ) As NoticiasConNumFila
        --Aquí le decimos que solo queremos que devuelva las filas que estén entre
        --los rangos que le hemos pasado por parámetro.
        WHERE RowRank > @startRowIndex AND RowRank <= (@startRowIndex +
@maximumRows)

END
```

Procedimientos de la tabla tb_Opinión.

Los procedimientos de las tablas que se describen a continuación, son muy similares a los de la tabla tb_Noticias, por tanto defina los procedimientos que se indican en cada una de las figuras. Se explicarán únicamente aquellos procedimientos que tengan variaciones con respecto a los anteriores o que sean totalmente diferentes.

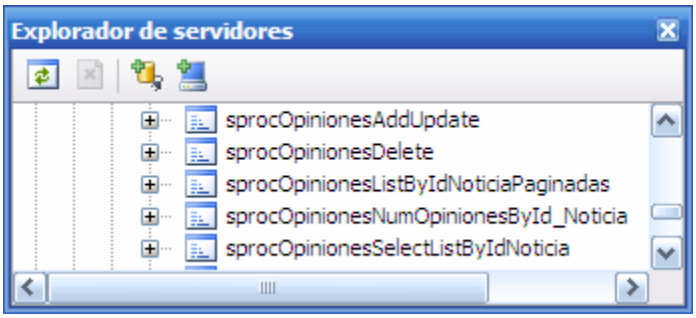


Fig. 42 Procedimientos de la tabla tb_Opinión.

sprocOpinionesSelectListByIdNoticia: permitirá conocer las opiniones asociadas a una noticia cuyo Id_Noticia coincida con el pasado como parámetro, estás deberán ser ordenadas descendentemente por el campo Fecha.

sprocOpinionesNumOpinionesById_Noticia : permitirá obtener el número de opiniones de una noticia cuyo Id_Noticia coincida con el valor del parámeto pasado, el resultado lo devolverá en un nuevo campo llamado numopiniones.

```
ALTER PROCEDURE dbo.sprocOpinionesNumOpinionesById_Noticia

        @id_noticia int
        AS
                SELECT    COUNT(1) AS numopiniones
                FROM      tb_Opinión
                WHERE    (Id_Noticia = @id_noticia)
```

sprocOpinionesListByIdNoticiaPaginadas: obtendrá las opiniones relacionadas con una detrminda noticia de forma paginada. El parámetro id, representa el Id de la noticia cuyas opiones se desean obtener, el número máximo de opiniones a obtener está representado por el parámetro maximunRows y la fila de inicio de esta selección lo indica el parámetro startRowIndex.

```
ALTER PROCEDURE dbo.sprocOpinionesListByIdNoticiaPaginadas

@startRowIndex int,
@maximumRows int,
@id int

AS
BEGIN
SET NOCOUNT ON;
        SELECT    Id_Opinion, Id_Noticia, Usuario, Comentario, Fecha, RowRank
        FROM
        (
        SELECT    Id_Opinion, Id_Noticia, Usuario, Comentario, Fecha, (ROW_NUMBER()
OVER (ORDER BY Fecha DESC)) AS RowRank
        FROM     tb_Opinión
        WHERE    (Id_Noticia = @id)

        ) As OpinionesConNumFila

        WHERE RowRank > @startRowIndex AND RowRank <= (@startRowIndex +
@maximumRows)

END
```

Procedimientos de la tabla tb_Encuesta.

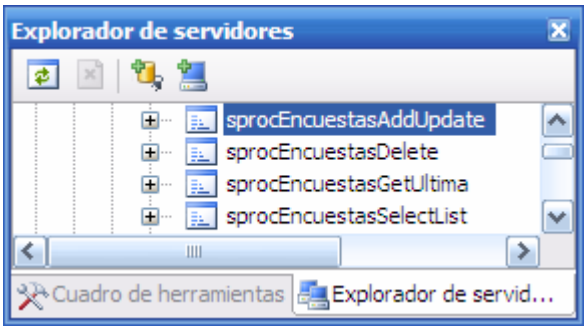


Fig. 43 Procedimientos de la tabla tb_Encuesta.

Los procedimientos almacenados de la tabla tb_Encuesta, son similares a los antes descritos, con excepción del siguiente procedimiento. Recuerde eliminar en el procedimiento almacenado sprocEncuestasDelete, los registros relacionados con la encuesta que se está eliminando en las tablas tb_RespuestaEncuesta y tb_RepuestaEncuestaUsr.

sprocEncuestasGetUltima: Procedimiento que permite obtener la última encuesta publicada en el portal.

Realiza una consulta sobre la tabla tb_Encuesta para seleccionar aquellos registros cuyo campo Activa sea igual a uno y Fecha_Cierre tenga un valor mayor al valor recibido como parámetro. De aquellos registros que cumplan dichos requisitos se seleccionará aquel cuyo campo Fecha_Publicación sea el más reciente, cláusula TOP.

```
ALTER PROCEDURE dbo.sprocEncuestasGetUltima
        @fecha datetime

        AS

                SELECT TOP (1) *
                FROM tb_Encuesta
                WHERE (Activa = 1) AND (Fecha_Cierre > @fecha)
                ORDER BY Fecha_Publicación DESC
```

Procedimientos de la tabla tb_RespuestaEncuesta

Recuerde en el procedimiento sprocRespuestasEncuestasDelete, eliminar también los registros relacionados en la tabla tb_RepuestaEncuestaUsr.

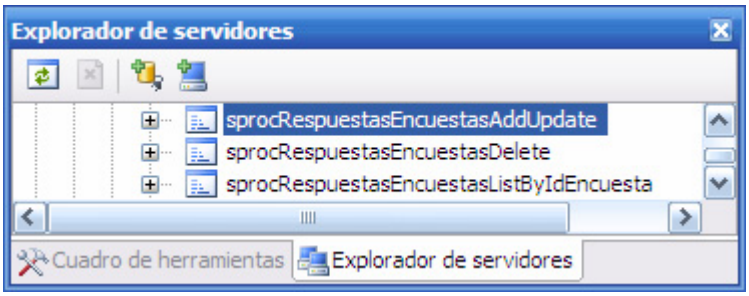


Fig. 44 Procedimientos de la tabla tb_RespuestaEncuesta.

sprocRespuestasEncuestasListByIdEncuesta: Procedimiento que permite obtener las respuestas a una determinada encuesta publicada.

Realiza una consulta sobre la tabla tb_RespuestaEncuesta, seleccionando todos los campos de aquellos registros cuyo Id_Encuesta coincida con el valor pasado como parámetro.

```
ALTER PROCEDURE dbo.sprocRespuestasEncuestasListByIdEncuesta
        @id_encuesta int

        AS
                SELECT * FROM
                tb_RespuestaEncuesta
                WHERE (Id_Encuesta = @id_encuesta)
```

Procedimientos de la tabla tb_RespuestaEncuestaUsr

sprocRespuestasEncuestassUsrSelectNumVotosByIdRespuesta: Obtiene el número de votos que ha recibido una determinada respuesta a una encuesta.

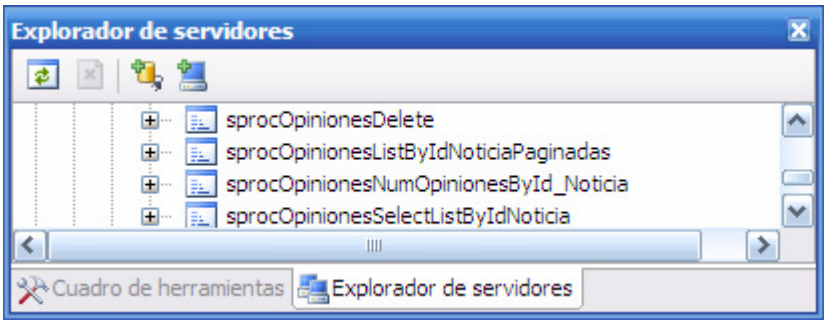


Fig. 45 Procedimientos de la tabla tb_RespuestaEncuestaUsr.

Realiza una consulta sobre la tabla tb_RespuestaEncuestaUsr, seleccionando los registros cuyo campo Id_Respuesta coindica con el pasado como parámetro, al incluir la cláusula COUNT, se cuenta el número de registros seleccionados y se retorna este valor en un nuevo campo llamado numvotos.

```
ALTER PROCEDURE dbo.sprocRespuestasEncuestasUsrSelectNumVotosByIdRespuesta

@id_respuesta int
AS
        SELECT    COUNT(1) AS numvotos
        FROM       tb_RespuestaEncuestaUsr
        WHERE    (Id_Respuesta = @id_respuesta)
```

sprocRespuestasEncuestasUsrHaVotado: Procedimiento que nos permite determinar si un usuario ha votado sobre una determinada encuesta.

Realiza una consulta sobre la tabla tb_RespuestaEncuestaUsr seleccionando los registros cuyos campos Id_Encuesta y Usuario, coincidan con los argumentos recibidos.

```
ALTER PROCEDURE dbo.sprocRespustasEncuestasUsrHaVotado
        @id_encuesta int,
        @usuario varchar(50)

        AS
                SELECT * FROM
                tb_RespuestaEncuestaUsr
                WHERE (@id_encuesta = Id_Encuesta AND @usuario = Usuario)
```

Procedimientos de la tabla tb_Foro

Recuerde en el procedimiento sprocForosDelete, eliminar también en la tabla tb_Mensaje, los registros cuyo Id_Foro coincida con el Id del foro que se está eliminando y que es pasado como parámetro.

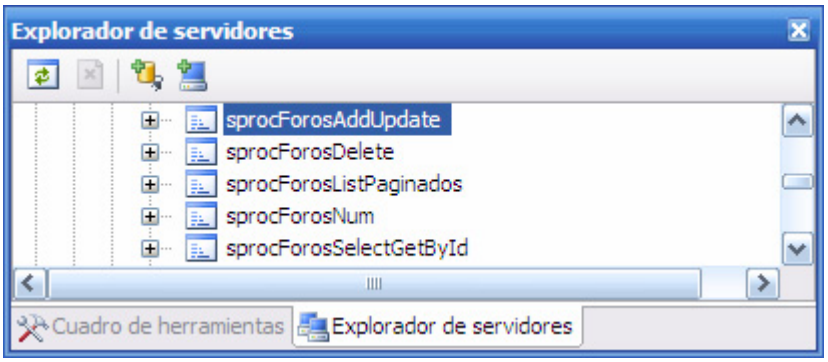


Fig. 46 Procedimientos de la tabla tb_Foro.

sprocForosListPaginados: Procedimiento que permite obtener un serie de foros paginados, realiza una consulta sobre la tabla tb_Foro obteniendo todos los campos de los registros, de esta selección, escoge los registros o filas que estén dentro del rango pasado como parámetro, en el cual startRowIndex, indica el punto de inicio de la selección y maximumRows el tamaño máximo de la selección, a partir del punto de inicio.

```
ALTER PROCEDURE dbo.sprocForosListPaginados

        @startRowIndex int,
        @maximumRows int
AS
        SELECT  *
        FROM
        (SELECT     *,  ROW_NUMBER() OVER (ORDER BY Nombre) AS RowRank

        FROM     tb_Foro
        )As ForosConNumFila

        WHERE RowRank > @startRowIndex AND RowRank <= (@startRowIndex +
@maximumRows)
```

sprocForosNum: Mediante este procedimiento sabremos el número de Foros publicados en el Portal. Realiza una consulta sobre la tabla tb_Foro, con la cláusula COUNT, cuenta el número de registros de la selección.

```
ALTER PROCEDURE dbo.sprocForosNum

        AS
                SELECT    COUNT(1) AS numeroforos
                FROM      tb_Foro
```

Procedimientos de la tabla tb_Mensaje

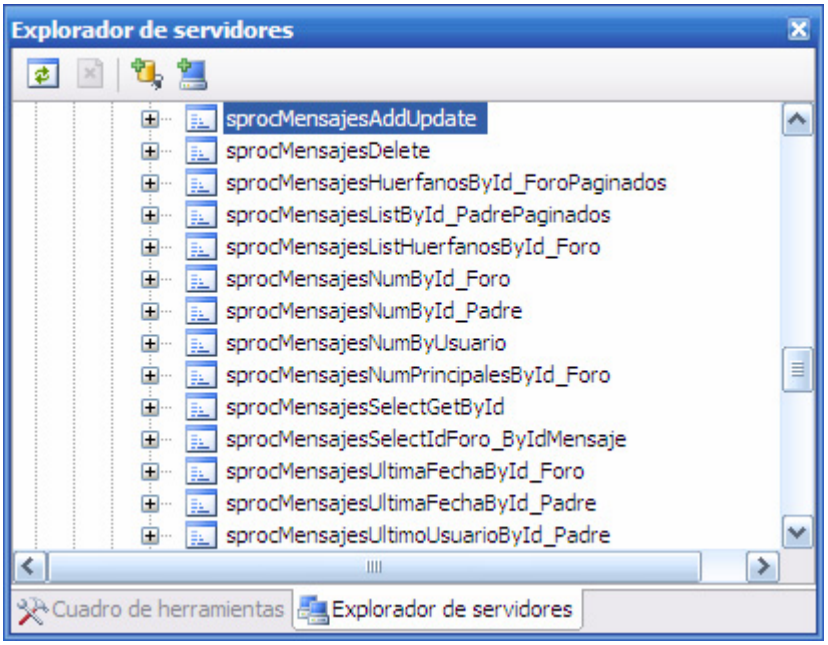


Fig. 47 Procedimientos de la tabla tb_Mensaje.

sprocMensajesHuerfanosById_ForoPaginados. Permite obtener una lista de mensajes en forma paginada. Realiza una consulta sobre la tabla tb_Mensaje, seleccionando aquellos registros cuyo campo Id_Foro coincide con el id pasado como parámetro y que no tienen padre, es decir mensajes que no dependen de otro (mensajes principales), de este resultado selecciona únicamente los registros o filas que estén en el rango especificado por los parámetros startRowIndex y maximunRows.

```
ALTER PROCEDURE dbo.sprocMensajesHuerfanosById_ForoPaginados
        @startRowIndex int,
        @maximumRows int,
        @id_foro int

        AS

                SELECT    *
                FROM
                        (
```

```
                    SELECT   *, ROW_NUMBER() OVER (ORDER BY Fecha) AS
                    RowRank
                    FROM     tb_Mensaje
                    WHERE     (Id_Foro = @id_foro) AND (Id_Padre IS NULL)
                    )As MensajesPrincipalesConNumPagina
            WHERE RowRank > @startRowIndex AND RowRank <= (@startRowIndex +
            @maximumRows)
```

sprocMensajesListById_PadrePaginados: Permite obtener, una lista de mensajes, realizando una consulta sobre la tabla tb_Mensaje y seleccionando los registros cuyos campos Id_Padre y Id_Mensaje coninciden con el Id pasado como parámetro, luego realiza un filtro sobre todos los registros obtenidos, para quedarse únicamente con aquellos que están en el rango especificado por los parámetros startRowIndex y masimumRows

```
ALTER PROCEDURE dbo.sprocMensajesListById_PadrePaginados
        @startRowIndex int,
        @maximumRows int,
        @id_padre int
AS

SELECT *
FROM  (

                SELECT *, ROW_NUMBER() OVER (ORDER BY Fecha) AS RowRank
                FROM tb_Mensaje
                WHERE (Id_Padre = @id_padre)  OR (Id_Mensaje = @id_padre)
                )As MensajesConNumFila

        WHERE RowRank > @startRowIndex AND RowRank <= (@startRowIndex +
        @maximumRows)
```

sprocMensajesListHuerfanosById_Foro: Selecciona todos los registros cuyo campo Id_Padre es NULL, además su campo Id_Foro deberá coincidir al valor pasado como parámetro, es decir obtiene la lista de todos los mensajes principales, mensajes que no dependen de otro.

sprocMensajesNumById_Foro: Permite conocer el número de mensajes que existen en un determinado Foro.

Realiza una consulta sobre la tabla tb_Mensaje, seleccionando aquellos registros cuyo campo Id_Foro coincida con el valor pasado como parámetro, de esta selección cuenta las ocurrencias, o el número de registros.

```
ALTER PROCEDURE dbo.sprocMensajesNumById_Foro
        @id_foro int

        AS
                SELECT    COUNT(1) AS nummensajes
                FROM      tb_Mensaje
                WHERE    (Id_Foro = @id_foro)
```

sprocMensajesNumById_Padre: Permite obtener el número de mensajes cuyo Id de padre coincide con el id pasado como parámetro, semejante al procedimiento anterior.

sprocMensajesNumByUsuario: Obtiene el número de mensajes escritos por el usuario pasado como parámetro. Semejante al procedimiento anterior.

sprocMensajesNumPrincipalesById_Foro: Este mensaje permite obtener el número de mensajes principales que tiene un determinado foro. Obtiene el número de mensajes cuyo campo Id_Foro coincide con el pasado como parámetro y su campo Id_Padre es NULL.

sprocMensajesSelectIdForo_ByIdMensaje: Procedimiento que realiza una consulta sobre la tabla tb_Mensaje, seleccionando únicamente el campo Id_Foro, de aquellos registros cuyo campo Id_Mensajes es igual al pasado como parámetro.

sprocMensajesUltimaFechaById_Foro: A través de este mensaje obtendremos la última fecha en la cual se ha introducido un mensaje a un determinado Foro.

Realiza una consulta sobre la tabla tb_Mensaje, seleccionando todos los registros cuyo Id_Foro, coincida con el pasado como parámetro, este resultado lo ordena de forma descendente mediante el campo Fecha, luego selecciona los campo Id_Foro y Fecha del primer registro, esto mediante la cláusula TOP.

```
ALTER PROCEDURE dbo.sprocMensajesUltimaFechaById_Foro
        @id_foro int

        AS
                SELECT    TOP (1) Id_Foro, Fecha
                FROM      tb_Mensaje
                WHERE    (Id_Foro = @id_foro)
                ORDER BY Fecha DESC
```

sprocMensajesUltimaFechaById_Padre: Procedimiento utilizado para obtener la última fecha en la que se ha introducido un mensaje (Submensaje) a un determinado mensaje padre, es semejante al procedimiento anterior.

sprocMensajesUltimoUsuarioById_Padre: Permite obtener el último usuario que ha escrito un mensaje en un determinado Foro.

Realiza una consulta sobre la tabal tb_Mensaje, seleccionando los registros cuyo Id_Padre coincida con el pasado como parámetro, luego los ordena de forma descendente con respecto al campo fecha, y selecciona los campos Usuario y Fecha del primer registro.

```
ALTER PROCEDURE dbo.sprocMensajesUltimoUsuarioById_Padre
        @id_padre int

        AS

        SELECT    TOP (1) Usuario, Fecha
        FROM       tb_Mensaje
        WHERE    (Id_Padre = @id_padre)
                ORDER BY Fecha DESC
```

Procedimientos de la tabla tb_CanalRSS.

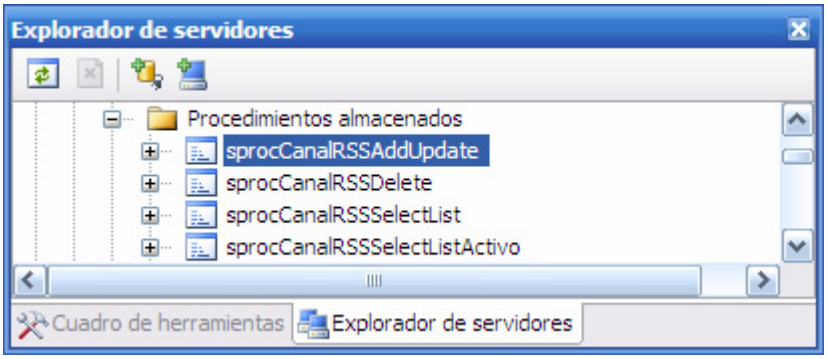


Fig. 48 Procedimientos de la tabla tb_CanalRSS.

sprocCanalRSSSelectList: Procedimiento que permite obtener el listado de todos los canales RSS del portal, realiza una consulta sobre la tabla tb_CanalRSS, obtiene todos los registros existentes, los ordena por el campo Título.

```
ALTER PROCEDURE dbo.sprocCanalRSSSelectList

        AS

        SELECT * FROM
        tb_CanalRSS
                ORDER BY Título DESC
```

sprocCanalRSSSelectListActivo: Permitirá obtener la lista de todos los canales RSS que se encuentran activos en el Portal. Realiza una consulta sobre la tabla tb_CanalRSS, seleccionado todos los campos de aquellos registros cuyo campo Activo sea igual a 1, ordenándolos con respecto al campo Título de forma ascendente.

```
ALTER PROCEDURE dbo.sprocCanalRSSSelectListActivo
        AS
                SELECT    *
                FROM        tb_CanalRSS
                WHERE Activo = 1
                ORDER BY Título
```

Procedimientos de la tabla tb_Enlace

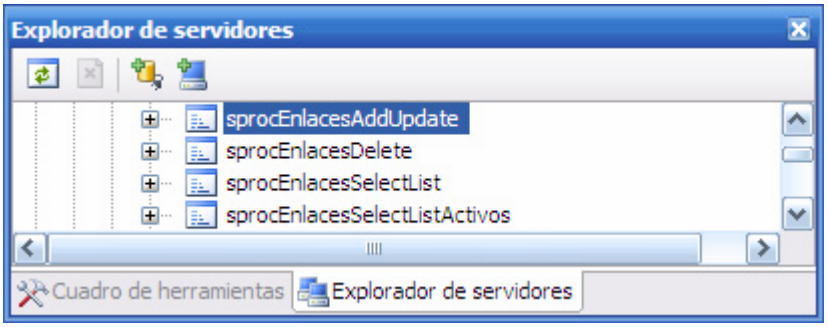


Fig. 49 Procedimientos de la tabla tb_Enlace.

sprocEnlacesSelectListActivos: obtiene todos los registros de la tabla tb_Enlace cuyo campo Activo tenga el valor de 1 (true).

Procedimientos de la tabla tb_Evento

sprocNumTotal: Procedimiento que permite obtener el número de eventos que se han publicado en el portal, realiza un consulta sobre la tabla tb_Evento, y realice una cuenta de los registros obtenidos, mediante la cláusula COUNT.

```
ALTER PROCEDURE dbo.sprocEventosNumTotal

AS
        SELECT    COUNT(*) AS total
                FROM       tb_Evento
```

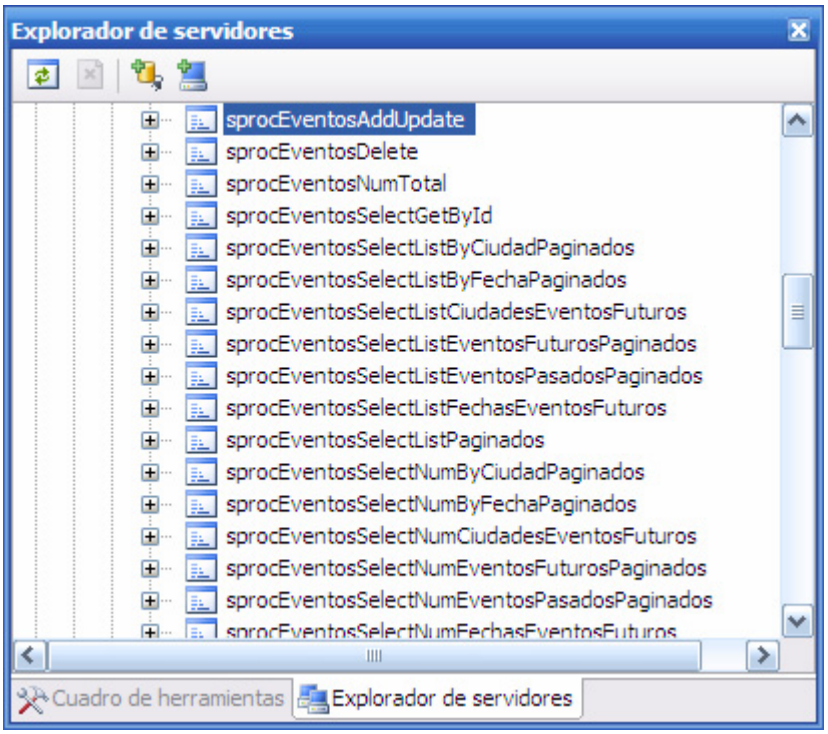


Fig. 50 Procedimientos de la tabla tb_Evento.

sprocEventosSelectListByCiudadesPaginados: Procedimiento mediante el cual obtenemos los eventos que se realizarán en una determinada ciudad, de forma paginada.

Realiza una consulta sobre la tabla tb_Evento, seleccionando todos los eventos cuyo campo Ciudad coincide con el pasado como parámetro y que se vayan a realizar el día actual o en un futuro, es decir su campo Fecha es mayor o igual al la fecha pasada como parámetro (fecha actual). De esta selección filtra únicamente aquellos registros que se encuentren en rango indicado por los parámetros startRowIndex y maximumRows

```
ALTER PROCEDURE dbo.sprocEventosSelectListByCiudadPaginados

@startRowIndex int,
@maximumRows int,
@ciudad varchar(MAX),
@hoy Datetime
```

```
AS

SELECT   *

FROM(
        SELECT    *, ROW_NUMBER() OVER (ORDER BY Fecha) AS RowRank
        FROM      tb_Evento
        WHERE     ((Ciudad=@ciudad) AND (Fecha >= @hoy))


        )As EventosConNumFila

WHERE RowRank > @startRowIndex AND RowRank <= (@startRowIndex +
@maximumRows)
```

sprocEventosSelectListByFechaPaginados: Procedimiento mediante el cual obtenemos los eventos que se realizarán en una determinada fecha, de forma paginada. Es similar al anterior, pero no se toma en cuenta el campo Ciudad y la fecha deberá coincidir con la pasada como parámetro.

sprocEventosSelectListCiudadesEventosFuturos: Procedimiento mediante el cual se obtiene la listas de la ciudades en las cuales se llevará acabo algún evento.

Realiza una consulta sobre la tabla tb_Evento, seleccionando únicamente el campo Ciudad (con la claúsula DISTINCT, se ignoran las repeticiones de los valores seleccionado) de aquellos registros cuyo campo Fecha sea mayor o igual a la pasada como parámetro, esta selección la ordena por el campo Ciudad y la retorna en un nuevo campo llamado EventosFuturos.

```
ALTER PROCEDURE dbo.sprocEventosSelectListCiudadesEventosFuturos

        @hoy DateTime
AS
        SELECT DISTINCT Ciudad
        FROM      (SELECT    *
                   FROM       tb_Evento
        WHERE     (Fecha >= @hoy)) AS EventosFuturos ORDER BY Ciudad
```

sprocEventosSelectListEventosFuturosPaginados: Permite obtener la lista de todos los eventos futuros de forma pagina, es similar al procedimiento sprocEventosSelectListByCiudadesPaginados, sin tomar en cuenta el campo Ciudad.

sprocEventosSelectListEventosPasadosPaginados: Permite obtener la lista de todos los eventos que se han llevado a cabo, por tanto ahora el campo Fecha deberá ser menor que el pasado como parámetro, similar al procedimiento sprocEventosSelectListByCiudadesPaginados.

sprocEventosSelectListFechasEventosFuturos. Este procedimiento es muy similar al sprocEventosSelectListCiudadesEventosFuturos, pero ahora seleccione el campo Fecha y ordene la selección también por este mismo campo.

sprocEventosSelectListPaginados: Similar al procedimiento sprocNoticiasListByIdNoticiaPaginadas, pero sin ninguna cláusula WHERE.

sprocEventosSelectNumByCiudadPaginados: Mediante este procedimiento se obtiene el número de ciudades en las cuales se llevarán eventos en el futuro o en el día actual.

Realiza una consulta sobre la tabla tv_Evento, devuelve en un nuevo campo llamado Total,el número de registos cuyo campo Ciudad coincide con el pasado como parámetro y que su campo Fecha es mayor o igual a la pasada como parámetro.

```
ALTER PROCEDURE dbo.sprocEventosSelectNumByCiudadPaginados

@ciudad varchar(MAX),
@hoy DateTime

AS

SELECT  COUNT (*) AS Total
FROM tb_Evento
        WHERE     ((Ciudad=@ciudad) AND (Fecha >= @hoy))
```

sprocEventosSelectNumByFechaPaginados: Similar al anterior, pero no se toma en cuanta el campo Ciudad y el campo Fecha deberá coincidir con el pasado como parámetro.

sprocEventosSelectNumCiudadesEventosFuturos: Obtiene el número de ciudades en las que se llevará a cabo algún evento.

Realiza un consulta sobre la tabla tb_Evento, contanto sin repeticiones, los valores del campo Ciudad, de aquellos registros cuyo campo Fecha es mayor o igual al pasado como parámetro.

```
ALTER PROCEDURE dbo.sprocEventosSelectNumCiudadesEventosFuturos

        @hoy DateTime
AS
        SELECT COUNT(DISTINCT Ciudad) AS Total
        FROM    (SELECT     *
                  FROM      tb_Evento
                  WHERE     (Fecha >= @hoy)) AS EventosFuturos
```

sprocEventosSelectNumEventosFuturosPaginados: Similar al procedimiento sprocEventosSelectNumByCiudadPaginados, pero sin tomar en cuenta el campo Ciudad.

sprocEventosSelectNumEventosPasadosPaginados: Similar al anterior, pero en esta ocación la Fecha deberá ser menor que el valor pasado como parámetro y el campo Review no deberá ser NULL.

```
ALTER PROCEDURE dbo.sprocEventosSelectNumEventosPasadosPaginados

@hoy DateTime

AS

SELECT  COUNT (*) AS Total
FROM tb_Evento

WHERE (Fecha < @hoy) AND (NOT (Review IS NULL))
```

sprocEventosSelectNumFechasEventosFuturos: Es similar al procedimiento sprocEventosSelectNumCiudadesEventosFuturos, pero la clúsula DISTINCT se ejecutará sobre el campo Fecha.

3. OBJETOS DE NEGOCIO

3.1. Introducción

Para intercambiar información entre la capa de acceso a datos y la de lógica de negocio, se utilizarán los objetos de negocio.

Un objeto de negocio es una clase que únicamente contiene propiedades, que se corresponderán en la mayoría de los casos con los campos que conforman cada tabla. Su misión principal es servir de contenedor destinado exclusivamente a la transferencia de información.

Cada objeto de negocio de la aplicación lleva parejo una clase destinada a almacenar una colección objetos de ese mismo tipo, en lo cual se aprovecharán las colecciones genéricas que nos proporciona .NET. Para una mejor organización del Portal estas todas estas clases se agruparán bajo un mismo espacio de nombre (*PortalWeb.BO*).

Preparar la carpeta **App_Code**:

- Clic derecho sobre el proyecto.
- Agregar carpeta ASP .NET
- Seleccionar App_Code (Aquí estará toda la capa de lógica de negocio y de acceso a datos, junto con los objetos de negocio.)
- Ahora cree una carpeta dentro de App_Code destinada al almacenamiento de los objetos de negocio, llamada *Business Object*.
- Para almacenar la colección de un objeto cree dentro de esta última carpeta, una nueva llamada *Collections*.

Para agregar un objeto, realice los siguientes pasos:

- Clic derecho sobre la carpeta *Business Object.*
- Agregar nuevo elemento ..
- Seleccione la plantilla **Clase**
- Asígnele nombre a la clase (*Noticia*)

Una vez creado el objeto de negocio (*Noticia*), proceda a editar dicha clase, incluya esta clase dentro de un espacio de nombre (*PortalWeb.BO*), se hará uso de las propiedades de las clases, declare una propiedad por cada campo de la tabla correspondiente al objeto de negocio en cuestión (en este caso la tabla tb_Noticias), para facilitar la localización y el mantenimiento del código usaremos las construcciones #region y #endregion, favoreciendo así la esquematización y claridad del código.

Dentro de una región se encontrarán aquellos campos privados de la clase, que coincidirán con los campos de la tabla, y dentro de otra región se encontrarán las propiedades que permiten establecer el valor de un campo privado o bien obtener el valor de dicho campo.

3.2. Objeto Noticia

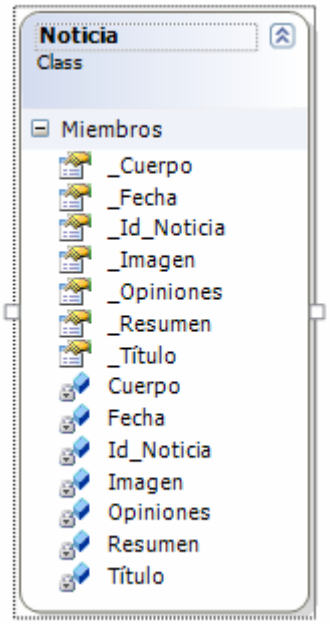


Fig. 51 Objeto de negocio: Noticia.

Realice los pasos necesarios para agregar una nueva clase (nuevo objeto) a la carpeta *Business Object.* A continuación se muestra el código completo de la clase Noticia.

```
namespace PortalWeb.BO
{
   public class Noticia
   {
        #region Variables Privadas
```

```
            private int Id_Noticia = -1;
            private String Título = String.Empty ;
            private String  Resumen = String.Empty;
            private String Cuerpo = String.Empty;
            private DateTime Fecha;
            private String Imagen = String.Empty;
            private ListOpinión  Opiniones = new ListOpinión ();
        #endregion

        #region Propiedades
            public int _Id_Noticia
            {
                get { return Id_Noticia; }
                set { Id_Noticia = value; }
            }

            public String _Título
            {
                get { return Título; }
                set { Título = value; }
            }

            public string _Resumen
            {
                get { return Resumen; }
                set { Resumen = value; }
            }

            public String _Cuerpo
            {
                get { return Cuerpo; }
                set { Cuerpo = value; }
            }

            public DateTime  _Fecha
            {
                get { return Fecha; }
                set { Fecha = value; }
            }

            public string  _Imagen
            {
                get { return Imagen; }
                set { Imagen = value; }
            }

            public ListOpinión _Opiniones
            {
                get { return Opiniones; }
                set { Opiniones = value; }
            }
        #endregion

    }
}
```

Como se observa en la región Variables Privadas, los campos aquí definidos corresponden en gran medida con las columnas de la tabla tb_Noticias de la base de datos. Cada uno de ellos inicializado a un valor por defecto. El campo Id_Noticia está inicializado a -1. Más tarde este valor se usará para determinar si un objeto es nuevo (el Id_Noticia es todavía -1), o si está asociado a un registro de la base de datos (el Id_Noticia es mayor que 0).

Los campos Título, Resumen, Cuerpo e Imagen tienen por defecto el valor *String.Empty*, señalando que aún no tienen contenido. El campo Opiniones es del tipo ListOpinión, es decir, da cabida a una colección de objetos de negocio del tipo Opinión. Esta lista nos permite acceder a la colección de opiniones asociada a una noticia.

Observe que las propiedades establecen el valor de su correspondiente campo privado mediante el método set y recuperan el valor del mismo mediante el método get.

Como se deduce de lo explicado anteriormente, la única misión las clases del espacio de nombres BO es contener datos, por lo que para trabajar con instancias de la clase Noticia necesitamos usar los métodos de la clase correspondiente en la capa de lógica de negocio (*BLLNoticia*) o los métodos de las clase correspondiente en la capa de acceso a datos (*DALNoticia*).

3.3. Colección de objetos de tipo Noticia

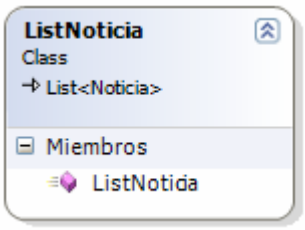


Fig. 52 Colección de objeto de negocio: ListNoticia.

C# incluye en su biblioteca de clases el espacio de nombres ***System.Collections.Generic*** que contiene interfaces y clases que definen tipos genéricos. Estos permiten al usuario crear colecciones particularizadas para un mismo tipo específico de datos. Aprovechando esta característica y la necesidad de manejar conjuntos de datos del mismo tipo, cada objeto de

negocio vendrá acompañado de una clase destinada a almacenar una colección de dicho objeto.

Podemos implementar las colecciones como tipos genéricos *List*<*T*> donde T hace referencia al tipo genérico del que se compondrá la lista, por ejemplo en el caso de Noticia, *List* <*Noticia*>, o en el caso de Comentario *List* <*Comentario*>. Las listas genéricas funcionan de manera similar a un array, permitiéndonos añadir y eliminar elementos de la lista de forma sencilla, a través de los métodos Add y Remove respectivamente.

Añada una nueva clase a la carpeta Collections definida anteriormente y asígnele un nombre en este caso *ListNoticia*, luego edítele para que se derive de una colección de tipo Noticia (objeto de negocio).

A continuación se muestra el código de la clase *ListNoticia*:

```
using System.Collections.Generic;

namespace PortalWeb.BO
{
   public class ListNoticia : List<Noticia>
   {
      public ListNoticia()
      {
                  // TODO: Agregar aquí la lógica del constructor
       }
   }
}
```

3.4. Diagrama de clase de los objetos de negocios

La muestra el diagrama de clases de todos los objetos de negocio, junto con su correspondiente colección, por tanto cree de manera similar al objeto Noticias y su colección, cada uno de los objetos que se muestran en el diagrama.
Para crear este diagrama una vez, que tenga definida todas las clases que se muestran en la **Fig. 53**, siga los siguientes pasos:

- Cree una subcarpeta dentro de la carpeta App_Code y nómbrela Diagram Class.
- Clic derecho sobre la carpeta creada en el punto anterior.
- Agregar nuevo elemento …
- Seleccione la plantilla Diagrama de clase
- Asígnele nombre, en este caso: Business Object.
- Clic en Agregar
- Desde el Explorador de soluciones, arrastre todas las clases de la carpeta Busines Object y de la subcarpeta Collections.
- Personalice el diagrama, de tal forma que sean visibles todas las clases.

Como se muestra en la **Fig. 53**, se ha declarado una clase con su correspondiente colección por cada tabla de la base de datos, recuerde que estas clases tienen como miembros privados variables que se corresponden con los campos de las tablas y tiene sus correspondientes propiedades públicas.

La clase Encuesta tiene un campo privado denominado Respuestas de tipo *ListRespuestaEncuesta,* defina en la clase Foro, un campo privado *Mensajes* de tipo *ListMensaje* y también defínalo en la clase Mensaje, en la clase RespuestaEncuesta define el campo *RespuestasUsr* de tipo *ListRespuestaEncuestaUsr.* Recuerde también definir cada una de las propiedades públicas de estos campos.

Fig. 53 Diagrama de Objetos de Negocio.

3.5. Peculiaridades del resto de objetos de negocios.

El objeto FuenteRSS e ItemFuenteRSS no se corresponden con ninguna tabla de la base de datos. El objeto FuenteRSS está destinado a albergar la información correspondiente a un determinado canal que nos proporcionará su objeto CanalRSS. Mientras que el objeto ItemFuenteRSS corresponderá con la información de cada uno de lo elementos de dicho canal. A continuación podemos observar los campos de ambos objetos:

```
public class FuenteRSS
   {
      #region Variables Privadas

      private String title = String.Empty;
      private String link = String.Empty;
      private String description = String.Empty;
      private String imageurl = String.Empty;
      private DateTime lastBuildDate = DateTime.Now;
      private String fullxml = String.Empty;
      private TipoRSS tiporss = TipoRSS.RSS20;
      private ListItemFuenteRSS items = new ListItemFuenteRSS();

      #endregion

      Propiedades
```

El objeto FuenteRSS tiene un campo privado denominado Items del tipo ListItemFuenteRSS, es decir una lista de objetos del tipo ItemFuenteRSS. Además, dispone de un campo denominado Tiporss cuyos valor corresponderá con alguno de los posibles dentro de la enumeración TipoRSS.

```
public enum TipoRSS { Atom, RSS10, RSS20, RSS091 }

   public class ItemFuenteRSS
   {
      #region Variables Privadas

      private string title;
      private string link;
      private string description;
      private string pubDate;

      #endregion

      Propiedades
```

Estas dos clases junto con el tipo enumerado, están declaradas en el mismo fichero que la clase CanalRSS.

4. CAPA DE ACCESO A DATOS (DAL – Data Access Layer)

4.1. Introducción

La capa de acceso a datos, es la capa que interactúa con la fuente de datos (base de datos), su diseño está basando en la creación de una clase por cada tabla en la base de datos, dentro de cada clase se deberá definir un método o función por cada procedimiento almacenado relacionado con la tabla en cuestión. Para una mayor organización estas clases serán agrupadas en un mismo espacio de nombres (*PortalWeb.DAL*).

Para añadir una clase DAL al Portal realice los siguientes pasos:

- Seleccionar App_Code (creada anteriormente)
- Ahora cree una carpeta dentro de App_Code destinada al almacenamiento de las clases DAL, llamada *Data Access*
- Clic derecho sobre la carpeta *Data Access.*
- Agregar nuevo elemento ..
- Seleccione la plantilla **Clase**
- Asígnele nombre a la clase (*DALNoticia*)

4.2. Diagrama de clases de la capa DAL

La **Fig. 54** muestra el diagrama de clases de la capa DAL, por tanto deberá crear cada una de las clases que se muestran en el diagrama. Para crear este diagrama una vez añadida todas las clases necesarias, deberá seguir los mismo pasos que en el creación del diagrama *Business Object* y añadir desde le *Explorador de soluciones* todas las clases de la carpeta *Data Access*.

Fig. 54 Diagrama de clases de la capa DAL.

Cada unas de estas clases deberá tratar con objetos de negocio, cuyos atributos constituirán los parámetros con los que trabajarán los procedimientos almacenados, para poder acceder a los objetos de negocio, se deberá hacer visible el espacio de nombre *PortalWeb.BO* con la cláusula *using*, esto en cada una de las clases:

```
using PortalWeb.DAL;
```

En la **Fig. 54** aparece una clase llamada ***DALConfiguration***, que no se corresponde con ninguna de las tablas de la base de datos, esta clase contendrá únicamente la propiedad ***ConnecionString***, la cual extrae la cadena de conexión a la base de datos ubicada en el fichero *web.config.*

Según lo indicado anteriormente, añada la clase *DALConfiguration* a la carpeta Data Access y defínela de la siguiente manera.

```
namespace PortalWeb.DAL
{
   public class DALConfiguration
   {
      #region Propiedad Pública

      public static String ConnectionString
      {
         //Obtener la cadena de conexión declarada en el archivo web.config
         get {
            return ConfigurationManager.ConnectionStrings["Base_Datos"].ConnectionString;
         }
      }
      #endregion
   }
}
```

Esta clase permite cambiar la base de datos, con tan sólo modificar la propiedad *ConnectionString* del archivo de configuración, por tanto defina esta propiedad en el archivo w*eb.config* de la siguiente manera:

```
<connectionStrings>
        <add name="Base_Datos" connectionString="Data
Source=(local)\SqlExpress;AttachDbFilename=|DataDirectory|\BD_DATOS.mdf;Integrated
Security=True;User Instance=True" providerName="System.Data.SqlClient"/>
</connectionStrings>
```

Como se observa en código anterior, se hace referencia a la base de datos creada anteriormente BD_DATOS.mdf

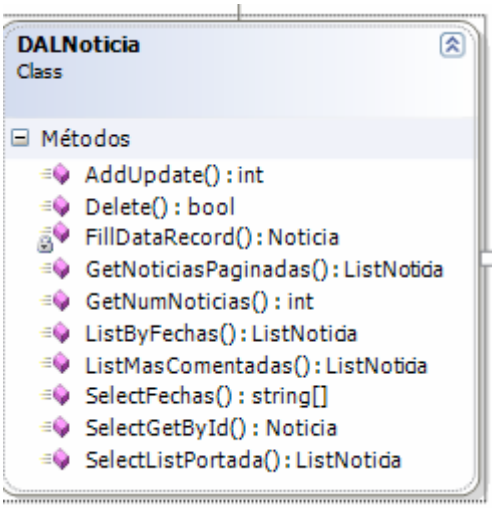
4.3. Organización del código de una clase DAL.

Para facilitar la localización y el mantenimiento del código se seguirá utilizando la definición de regiones, en la **Fig. 55** se muestra la organización que seguirán todas las clases de la capa DAL, organización que deberá tener en cuenta en cada una de las definiciones de las clases mostradas la **Fig. 54**.

```
using System.Data.Common;
using System.Data.SqlClient;
using PortalWeb.BO;

namespace PortalWeb.DAL
{

    public class DALNoticia
    {
        Métodos Públicos

        Métodos Privados

    }
}
```

Fig. 55 Organización de las clases de la capa DAL.

4.4. Clase DALNoticia



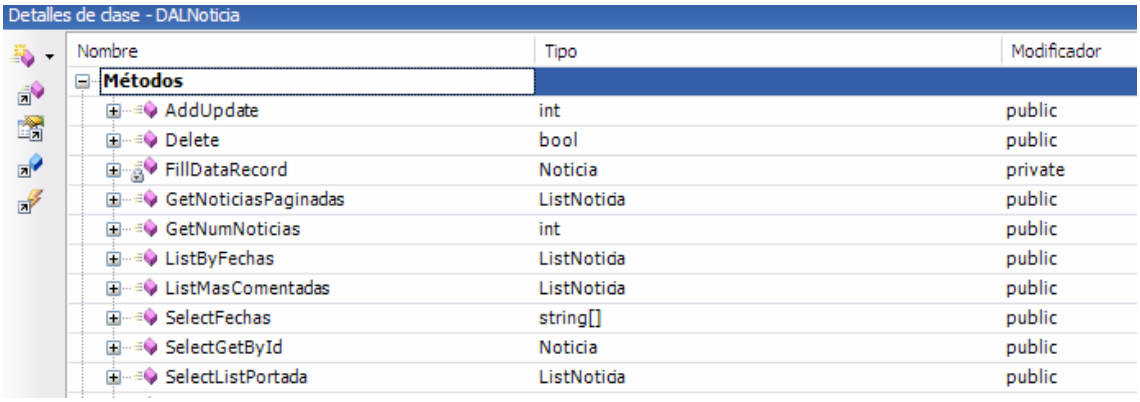


Fig. 56 Clase DALNoticia.

Las figuras anteriores muestran los detalles de la clase *DALNoticia*, como se observa en la **Fig. 56** aparece primero el nombre del método y a continuación separado con dos puntos el de valor de retorno, por tanto a la hora de definir el resto de clases, deberá observar estos métodos en la **Fig. 54**. La parte inferior de la **Fig. 56** es simplemente una ampliación de la clase, observe que todos los métodos son públicos excepto el método FillDataRecord, esto mismo se aplica al resto de clases.

Una vez creada la clase DALNoticia, proceda a definir los métodos mostrados en la de dicha clase.

4.4.1. Método Delete

```
public static bool Delete(int id)
{
    int re = 0;
    SqlConnection conexion = new SqlConnection(DALConfiguration.ConnectionString);
    SqlCommand orden = new SqlCommand("sprocNoticiasDelete", conexion);
    orden.CommandType = CommandType.StoredProcedure;
    orden.Parameters.AddWithValue("@id_noticia", id);
    conexion.Open();
    re = orden.ExecuteNonQuery();
    conexion.Close();
    return re > 0;
}
```

Primero establece una conexión al origen de datos, empleando un objeto SqlConnection, como parámetro recibe la cadena de conexión, extraída mediante la propiedad ConnectionString de la clase DALConfiguration, del fichero web.config.

Luego se crea un objeto SqlCommad, para ejecutar un procedimiento almacenado sobre la base de datos, recibe como parámetro el nombre del procedimiento a ejecutar "sprocNoticiasDelete" y el objeto conexión. Para indicar que se va a ejecutar un procedimiento almacenado, se asigna a la propiedad CommandType del objeto orden el valor de CommandType.StoredProcedure.

A continuación se pasan los parámetros requeridos por el procedimiento almacenado a ejecutar, para ello se hace uso del método AddWithValue de la propiedad Parameters del objeto orden, este método recibe el nombre del parámetro (nombre definido en el procedimiento almacenado, en este caso @id_noticia), y el valor asignado al parámetro.

Finalmente se abre la conexión a la base de datos, se ejecuta la orden (en este caso procedimiento almacenado), con el método ExecuteNonQuery, que retorna el número de filas afectadas, se cierra la conexión y se devuelve verdadero en caso de que se haya afectado alguna fila (la operación re realizó correctamente) o falso en caso contrario.

4.4.2. Método SelectGetById

```
public static Noticia SelectGetById(int id)
{
    Noticia minoticia = null;
    SqlConnection conexion = new SqlConnection(DALConfiguration.ConnectionString);
    SqlCommand orden = new SqlCommand("sprocNoticiasSelectGetById", conexion);
    orden.CommandType = CommandType.StoredProcedure;
    orden.Parameters.AddWithValue("@id_noticia", id);
    conexion.Open();
    SqlDataReader lector = orden.ExecuteReader();
    if (lector.Read())
        minoticia = FillDataRecord(lector);
    lector.Close();
    conexion.Close();
    return minoticia;
}
```

Este método obtendrá la noticia cuyo Id, reciba como parámetro. Primero se declara un objeto de negocio Noticia, luego al igual que en el método anterior, se crea una conexión y acto seguido se crea un objeto orden indicando el procedimiento almacenado a ejecutar, luego se indica que la orden es un procedimiento almacenado y se le pasan los argumento (@id_noticia), y se abre la conexión.

Esta vez se requiere realizar una lectura sobre origen de datos, por lo que se utilizará un objeto lector de datos o DataReader. Éste recuperará el conjunto de valores procedentes de la base de datos llenando un pequeño búfer de datos. Por tanto la orden se ejecutará con el método ExecuteReader, que devolverá un objeto de tipo DataReader. Seguidamente se llama al método Read del lector para leer los datos.

Cada fila en el SqlDataReader implementa la interfaz genérica IDataRecord que ofrece un acceso fuertemente tipado a los valores de cada fila. Dicha interfaz proporciona acceso a los valores de columna de cada fila para un DataReader. La implementan los proveedores de datos de .NET Framework que tienen acceso a bases de datos relacionales.

El método FillDataRecord, incluido en cada clase de la capa DAL, se utiliza para crear y rellenar una instancia de la clase Noticia basándose en los datos de la interfaz IDataRecord.

Finalmente se cierra el lector y la conexión, y se devuelve un objeto Noticia relleno con la información extraída de la base de datos.

4.4.3. Método FillDataRecord

```
private static Noticia FillDataRecord(IDataRecord myDataRecord)
{
  Noticia minoticia = new Noticia();
  minoticia._Id_Noticia = myDataRecord.GetInt32(myDataRecord.GetOrdinal("Id_Noticia"));
  minoticia._Título  = myDataRecord.GetString(myDataRecord.GetOrdinal("Título"));
  minoticia._Resumen = myDataRecord.GetString(myDataRecord.GetOrdinal("Resumen"));
  minoticia._Cuerpo = myDataRecord.GetString(myDataRecord.GetOrdinal("Cuerpo"));
  minoticia._Fecha = myDataRecord.GetDateTime(myDataRecord.GetOrdinal("Fecha"));
  if (!myDataRecord.IsDBNull(myDataRecord.GetOrdinal("Imagen")))
      minoticia._Imagen = myDataRecord.GetString(myDataRecord.GetOrdinal("Imagen"));
  return minoticia;
}
```

Este método será definido en cada una de las clases de la capa DAL. Crea una instancia de un objeto de negocio, en este caso un objeto Noticia. En cada campo de esta instancia introduce el valor correspondiente de la interfaz IDataRecord. Al final devuelve el objeto de negocio.

El código de este método simplemente copia los datos del lector en los campos privados del objeto de negocio haciendo uso de los métodos Get* del IDataRecord apropiados. Estos

métodos Get* reciben como parámetro el índice en base cero de la columna del IDataRecord. Para hacer más fácil la comprensión y el mantenimiento del código usaremos el método GetOrdinal para traducir el nombre de la columna de la tabla en su correspondiente índice.

Dentro de los métodos Get se utilizará: GetInt32 para enteros, GetString para cadenas de caracteres y GetDateTime para objetos del tipo DateTime. Cuando las propiedades son columnas que no admiten valores nulos en la base de datos no es necesario comprobar en este método si los valores son nulos o no. En los casos en los que podamos tener valores nulos tendremos que comprobar si efectivamente es así mediante el método IsDBNull del IDataRecord, y en ese supuesto dejar el valor por defecto declarado en los atributos.

4.4.4. Método SelectListPortada

```
public static ListNoticia SelectListPortada()
{
    ListNoticia listnoticia = null;
    SqlConnection conexion = new SqlConnection(DALConfiguration.ConnectionString);
    SqlCommand orden = new SqlCommand("sprocNoticiasSelectListPortada", conexion);
    orden.CommandType = CommandType.StoredProcedure;
    conexion.Open();
    SqlDataReader lector = orden.ExecuteReader();
     if (lector.HasRows)
     {
            listnoticia = new ListNoticia();
            while (lector.Read())
                listnoticia.Add(FillDataRecord(lector));
     }
     lector.Close();
     conexion.Close();
     return listnoticia;
}
```

Este método retorna una colección del objeto de negocio Noticia, esta colección representa las noticias de la base de datos que se van a presentar en la portada, es el procedimiento almacenado quién establece el filtro de entre todas las noticias. Primero se declara una colección de objetos Noticias que al final será el valor retornado por este método.

Luego se realiza todo lo explicado en los métodos anteriores, a continuación se ejecutar la orden con ExecuteReader, después de comprobar mediante la propiedad HasRows, que el objeto lector contiene una o más filas, se inicializa la colección. En cada iteración de un bucle se

crea una nueva instancia del tipo Noticia con el método FillDataRecord y a su vez se añade a la lista. Este bucle finalizará cuando no haya más filas en el lector, es decir, cuando la propiedad HasRows sea false.

4.4.5. Método ListByFechas

```
public static ListNoticia ListByFechas(DateTime fecha)
{
    DateTime fecha1;
    DateTime fecha2;
    fecha1 = fecha.Date;
    TimeSpan hora = new TimeSpan(23, 59, 59);
    fecha2 = fecha.Date.Add(hora);
    ListNoticia listNoticias = null;
    SqlConnection conexion = new SqlConnection(DALConfiguration.ConnectionString);
    SqlCommand orden = new SqlCommand("sprocNoticiasSelectListByFecha", conexion);
    orden.CommandType = CommandType.StoredProcedure;
    orden.Parameters.AddWithValue("@fecha1", fecha1);
    orden.Parameters.AddWithValue("@fecha2", fecha2);
    conexion.Open();
    SqlDataReader lector = orden.ExecuteReader();
    if (lector.HasRows)
    {
        listNoticias = new ListNoticia();
         while (lector.Read())
             listNoticias.Add(FillDataRecord(lector));
    }
    lector.Close();
    conexion.Close();
    return listNoticias;
}
```

Este método retornará aquellas noticias publicadas en la fecha que recibe como parámetro. Declara un objeto de tipo DataTime que inicializa mediante el método Date con el componente correspondiente a la fecha del parámetro recibido. Luego inicializa otro objeto DataTime y lo inicializa pero esta vez utilizando un objeto de tipo TimeSpan. Luego crea una instancia de una colección del objeto Noticia, valor que será retornado al final del método.

A continuación realiza todo el procedimiento descrito en los métodos anteriores, note que el procedimiento almacenado a ejecutarse recibe dos parámetros y por tanto se debe añadir cada uno por separado con el método AddWithValue indicando el nombre del parámetro y el valor.

4.4.6. Método ListMasComentadas

```
public static ListNoticia ListMasComentadas()
{
  int id;
  ListNoticia listnoticias = null;
  SqlConnection conexion = new SqlConnection(DALConfiguration.ConnectionString);
  SqlCommand orden = new SqlCommand("sprocNoticiasListMasComentadas", conexion);
  orden.CommandType = CommandType.StoredProcedure;
  conexion.Open();
  SqlDataReader lector = orden.ExecuteReader();
  if (lector.HasRows)
  {
     listnoticias = new ListNoticia();
     while (lector.Read())
     {
        id = Convert.ToInt32((lector["Id_Noticia"].ToString()));
        listnoticias.Add(SelectGetById(id));
      }
  }
  lector.Close();
  conexion.Close();
  return listnoticias;
}
```

Este método invoca al procedimiento almacenado encargado de obtener las 5 noticias que han recibido el mayor número de opiniones. En cada iteración del bucle de lectura se obtiene un identificador y luego se llama al método SelectGetById pasándole el mismo como parámetro. De esta manera se va llenando la lista de objetos de negocio del tipo Noticia. Para obtener el identificador no se hizo uso del método FillDataRecord. En este caso, le indica al objeto SqlDataReader el nombre de la columna que se desea leer (Id_Noticia). Como el valor retornado es del tipo object convertimos éste a tipo entero.

4.4.7. Método SelectFechas

```
public static string[] SelectFechas()
{
   string[] arrayfechas;
   int contador = 0, i = 0;
   SqlConnection conexion = new SqlConnection(DALConfiguration.ConnectionString);
   SqlCommand orden1 = new SqlCommand("sprocNoticiasSelectNumFechas", conexion);
   orden1.CommandType = CommandType.StoredProcedure;
   SqlCommand orden2 = new SqlCommand("sprocNoticiasSelectFechas", conexion);
   orden2.CommandType = CommandType.StoredProcedure;
   conexion.Open();
   SqlDataReader lector = orden1.ExecuteReader();
```

```
    if (lector.Read())
        contador = lector.GetInt32(0);
    arrayfechas = new string[contador];
    lector.Close();
    SqlDataReader lector2 = orden2.ExecuteReader();
    while (lector2.Read())
    {
        arrayfechas[i] = lector2.GetDateTime(0).ToShortDateString();
        i++;
    }
    lector2.Close();
    conexion.Close();
    return arrayfechas;
}
```

Este método retorna un array de tipo String con las fechas en las cuales se han publicado noticias. Lo primero se declara un array del tipo String y dos variables auxiliares del tipo int (i y contador). La primera servirá de índice en el array, y la segunda para almacenar el número de elementos de los que se compondrá dicho array.

A continuación se configuran dos objetos SqlCommand. El primero de ellos va a hacer uso del procedimiento almacenado que devuelve el número de fechas en las que se ha publicado alguna noticia. Mientras que el segundo obtiene dichas fechas.

Seguidamente se abre la conexión y se ejecuta la primera de las dos órdenes. En este caso el procedimiento almacenado únicamente devolverá un valor que recibimos utilizando el método GetInt32 del objeto SqlDataReader, ya que se trata de un entero. Dicho valor se le asigna a la variable contador y se inicializa el array con ese número de elementos. A continuación se ejecuta la segunda orden y en el bucle de lectura se va añadiendo al array las diferentes fechas que devuelve el lector. Finalmente se cierra el lector y la conexión. El valor retornado del método es el array relleno con la diferentes fechas en las que se han publicado noticias.

4.4.8. Método AddUpdate

```
public static int AddUpdate(Noticia noticia)
{
  int resultado = 0;
  SqlConnection conexion = new SqlConnection(DALConfiguration.ConnectionString);
  SqlCommand orden = new SqlCommand("sprocNoticiasAddUpdate", conexion);
  orden.CommandType = CommandType.StoredProcedure;
  if (noticia._Id_Noticia == -1)
```

```
        orden.Parameters.AddWithValue("@id_noticia", DBNull.Value);
    else
        orden.Parameters.AddWithValue("@id_noticia", noticia._Id_Noticia);
    orden.Parameters.AddWithValue("@titulo", noticia._Título);
    orden.Parameters.AddWithValue("@resumen", noticia._Resumen);
    orden.Parameters.AddWithValue("@cuerpo", noticia._Cuerpo);
    orden.Parameters.AddWithValue("@fecha", noticia._Fecha);
    if (String.IsNullOrEmpty(noticia._Imagen))
        orden.Parameters.AddWithValue("@imagen", DBNull.Value);
    else
        orden.Parameters.AddWithValue("@imagen", noticia._Imagen);

    DbParameter returnValue;
    returnValue = orden.CreateParameter();
    returnValue.Direction = ParameterDirection.ReturnValue;
    orden.Parameters.Add(returnValue);
    conexion.Open();
    orden.ExecuteNonQuery();
    resultado = Convert.ToInt32(returnValue.Value);
    conexion.Close();
    return resultado;
}
```

Este método tiene como finalidad insertar un nuevo registro o actualizar un registro existente en la base de datos, el registro a insertar o actualizar es el objeto de negocio Noticia que recibe como parámetro.

En primer lugar se declara variable de tipo int, que será el valor retornado por el método, es decir, el identificador del nuevo objeto creado o el identificador del objeto modificado. A continuación se crea el objeto orden y lo se configura con la conexión y el procedimiento almacenado correspondiente.

Posteriormente se configuran los parámetros que recibe el procedimiento almacenado. Estos parámetros se corresponden con las propiedades del BO homólogas a las columnas de la tabla tb_Noticias de la base de datos. Debido a que este método puede insertar o modificar, se debe comprobar si el campo Id_Noticia del objeto Noticias es -1, se trata de una inserción y por tanto se le pasa al parámetro procedimiento almacenado un valor NULL (DBNULL.Value), en caso contrario se inicializa con el valor del campo Id_Noticia del objeto de negocio, debido a que el campo Imagen de la tabla tb_Noticias acepta valores nulos, también se debe hacer esta comprobación. Finalmente se abre la conexión, se ejecuta la ordenmediante el método ExecuteNonQuery, se cierra la conexión y se retorna.

4.4.9. Método GetNoticiasPaginadas

```
public static ListNoticia GetNoticiasPaginadas(int startRowIndex, int maximumRows)
{
  ListNoticia listaTemporal = null;
  SqlConnection conexion = new SqlConnection(DALConfiguration.ConnectionString);
  SqlCommand orden = new SqlCommand("sprocNoticiasSelectListPaginadas", conexion);
  orden.Parameters.AddWithValue("@startRowIndex", startRowIndex);
  orden.Parameters.AddWithValue("@maximumRows", maximumRows);
  orden.CommandType = CommandType.StoredProcedure;
  conexion.Open();
  SqlDataReader lector = orden.ExecuteReader();
  if (lector.HasRows)
  {
     listaTemporal = new ListNoticia();
     while (lector.Read())
         listaTemporal.Add(FillDataRecord(lector));
  }
  lector.Close();
  conexion.Close();
  return listaTemporal;
}
```

Obtiene las noticias en forma paginada, primero se declara una colección de objetos Noticia que será el valor de retorno. Se añaden los parámetros requeridos por el procedimiento a almacenar y finalmente se ejecuta.

4.4.10. Método GetNumNoticias

```
public static int GetNumNoticias()
{
  int resultado = 0;
  SqlConnection conexion = new SqlConnection(DALConfiguration.ConnectionString);
  SqlCommand orden = new SqlCommand("sprocNoticiasSelectNumNoticias", conexion);
  orden.CommandType = CommandType.StoredProcedure;
  conexion.Open();
  SqlDataReader lector = orden.ExecuteReader();
  if (lector.HasRows)
      while (lector.Read())
          resultado = Convert.ToInt32((lector["numnoticias"].ToString()));
  lector.Close();
  conexion.Close();
  return resultado;
}
```

Retorna un entero que representa el número de noticias publicadas en el Portal. En el ciclo de lectura, se obtiene el número de noticias, se hace una conversión al tipo del valor retornado.

4.5. Peculiaridades del resto de las clases DAL.

Define los métodos se muestran en la **Fig. 54** para cada una de las clases DAL, a continuación se muestran los procedimientos que difieren de los ejemplares mostrados. Recuerde como forma genérica, declarar un objeto del tipo de valor de retorno del método en cuestión, crear el objeto conexión, el objeto orden y asociarle el procedimiento adecuado, indicar que la orden es de tipo procedimiento almacenado, añadir al procedimiento almacenado los parámetros que está esperando, abrir la conexión, ejecutar la orden, utilizar un objeto lector cuando proceda, cerrar la conexión y retornar el valor adecuado.

4.5.1. Clase DALEncuesta

Método SelectGetUltima

```
public static Encuesta SelectGetUltima(DateTime fecha){
  Encuesta miEncueta = null ;
  SqlConnection conexion = new SqlConnection(DALConfiguration.ConnectionString);
  SqlCommand orden = new SqlCommand("sprocEncuestasGetUltima", conexion);
  orden.CommandType = CommandType.StoredProcedure;
  orden.Parameters.AddWithValue("@fecha", fecha);
  conexion.Open();
  SqlDataReader lector = orden.ExecuteReader();
  if(lector.Read())
      miEncueta = FillDataRecord(lector);
  lector.Close();
  conexion.Close();
  return miEncueta;      }
```

4.5.2. Clase DALEvento

Método GetCiudadesFuturas

```
public static string[] GetCiudadesFuturas(DateTime hoy)
{
  string[] arrayciudades;
  int contador = 0;
  nt i = 0;
  SqlConnection conexion = new SqlConnection(DALConfiguration.ConnectionString);
  SqlCommand orden1 = new
SqlCommand("sprocEventosSelectNumCiudadesEventosFuturos", conexion);
  orden1.CommandType = CommandType.StoredProcedure;
```

```
  orden1.Parameters.AddWithValue("@hoy", hoy);
  SqlCommand orden2 = new
SqlCommand("sprocEventosSelectListCiudadesEventosFuturos", conexion);
  orden2.CommandType = CommandType.StoredProcedure;
  orden2.Parameters.AddWithValue("@hoy", hoy);
  conexion.Open();
  SqlDataReader lector = orden1.ExecuteReader();
  if (lector.Read())
  {
    contador = lector.GetInt32(0);
  }
  arrayciudades = new string[contador];
  lector.Close();
  SqlDataReader lector2 = orden2.ExecuteReader();
  while (lector2.Read())
  {
     arrayciudades[i] = lector2.GetString(0);
     i++;
  }
  lector2.Close();
  conexion.Close();
  return arrayciudades;
}
```

El método *GetFechasFuturas* es similar a este, con la diferencia de utilizar los procedimientos *sprocEventosSelectNumFechasEventosFuturos* y *sprocEventosSelectListFechasEventosFuturos* y que en el ciclo de lectura (*arrayciudades[i] = lector2.GetString(0)*), se deberá reemplazar por la siguiente sentencia (*arrayfechas[i] = lector2.GetDateTime(0).ToShortDateString()*), donde arraryfechas representa el valor a retornar.

Método GetEventosFuturosPaginados

```
public static ListEvento GetEventosFuturosPaginados(DateTime hoy, int startRowIndex, int
maximumRows)
{
  ListEvento listaTemporal = null;
  SqlConnection conexion = new SqlConnection(DALConfiguration.ConnectionString);
  SqlCommand orden = new
SqlCommand("sprocEventosSelectListEventosFuturosPaginados", conexion);
  orden.Parameters.AddWithValue("@hoy", hoy);
  orden.Parameters.AddWithValue("@startRowIndex", startRowIndex);
  orden.Parameters.AddWithValue("@maximumRows", maximumRows);
  orden.CommandType = CommandType.StoredProcedure;
  conexion.Open();
  SqlDataReader lector = orden.ExecuteReader();
  if (lector.HasRows)
  {
     listaTemporal = new ListEvento();
          while (lector.Read())
```

```
            {
                listaTemporal.Add(FillDataRecord(lector));
            }
        }
        lector.Close();
    conexion.Close();
        return listaTemporal;
    }
```

Este método recibe como argumente la fecha actual (fecha), el inicio de la selección (startRowIndex) para el procedimiento almacenado y el tamaño máximo (maximunRows) de esta selección, se declara una colección de objetos Noticia, objeto que será retornado al final de este método.
A continuación se crea la conexión, la orden, se pasan los argumentos esperados al procedimiento con sus valores, en este caso coinciden con los pasados como parámetros a este método, luego se hacen todas las operaciones que hasta aquí se han venido describiendo.

El método *GetEventosPasadosPaginados* sigue el mismo esquema pero esta vez utilizando el procedimiento sprocEventosSelectListEventosPasadosPaginados. Otro método que sigue el mismo procedimiento es *GetEventosByFechaPaginados*, pero en esta ocasión utilizando el procedimiento sprocEventosSelectListByFechaPaginados.

El método *GetEventosFuturosByCiudadPaginados*, también sigue el mismo esquema, pero recibe un cuarto parámetro de tipo string (ciudad), utiliza el procedimiento almacenado sprocEventosSelectListByCiudadPaginados y se deben añadir como parámetro además de fecha, starRowIndex y maximunRows, el parámetro ciudad.

Método GetNumEventosPaginados

```
public static int GetNumEventosPaginados()      {
  int total = 0;
  SqlConnection conexion = new SqlConnection(DALConfiguration.ConnectionString);
  SqlCommand orden = new SqlCommand("sprocEventosNumTotal", conexion);
  orden.CommandType = CommandType.StoredProcedure;
  conexion.Open();
  SqlDataReader lector = orden.ExecuteReader();
  if (lector.Read())
      total = lector.GetInt32(0);
  lector.Close();
  conexion.Close();
  return total;      }
```

Este procedimiento retorna un entero, que se corresponde con el número total de Eventos publicados en el portal. Declara una variable de tipo int, que será el valor de retorno, a continuación desarrolla un proceso muy similar a los otros métodos.

El método *GetNumEventosFuturosPaginados* sigue una estructura muy similar a la anterior, recibe como parámetro un objeto de tipo DataTime (hoy), utiliza como procedimiento almacenado sprocEventosSelectNumEventosFuturosPaginados, el parámetro recibido como argumento (Fecha), es añadido al procedimiento almacenado, el resto es igual. Ahora bien el método *GetNumEventosPasadosPaginados*, es igual a *GetNumEventosFuturosPaginados*, pero ahora utilizando el procedimiento sprocEventosSelectNumEventosPasadosPaginados. Otro evento que es igual a estos es *GetNumEventosFuturosByFechaPaginados*, pero utilizando el procedimiento sprocEventosSelectNumByFechaPaginados.

GetNumEventosFuturosByCiudadPaginados, es un método muy similar a los descritos en el párrafo anterior, pero ahora recibe a parte del parámetro fecha, otro de tipo string (ciudad), añade estos dos parámetros al procedimiento sprocEventosSelectNumByCiudadPaginados.

4.5.3. Clase DALForo

Método Foro_Num

```
public static int Foro_Num()
{
  int resultado = 0;
  SqlConnection conexion = new SqlConnection(DALConfiguration.ConnectionString);
  SqlCommand orden = new SqlCommand("sprocForosNum", conexion);
  orden.CommandType = CommandType.StoredProcedure;
  conexion.Open();
  SqlDataReader lector = orden.ExecuteReader();
  if (lector.HasRows)
      while (lector.Read())
  resultado = Convert.ToInt32((lector["numeroforos"].ToString()));
  lector.Close();
  conexion.Close();
  return resultado;
}
```

Este procedimiento, retorna un entero que se corresponde con el número total de Foros publicados en el Portal.

4.5.4. Clase DALMensaje

Método ListHuerfanosById_Foro

Dentro del foro, se puede distinguir entre los mensajes que dependen o subyacen de otros mensajes, y los mensajes principales. A la hora de recabar éstos últimos, hemos de hacer uso del método *ListHuerfanosById_Foro*. Para ello, se llama a la consulta correspondiente, pasándole como parámetro el identificador del mensaje. Obteniendo así una colección de mensajes ListMensaje con todos los mensajes principales listos para ser mostrados.

```
public static ListMensaje ListHuerfanosById_Foro(int id)
{
   ListMensaje listmensaje = null;
   SqlConnection conexion = new SqlConnection(DALConfiguration.ConnectionString);
   SqlCommand orden = new SqlCommand("sprocMensajesListHuerfanosById_Foro",
conexion);
   orden.CommandType = CommandType.StoredProcedure;
   orden.Parameters.AddWithValue("@id_foro", id);
   conexion.Open();
   SqlDataReader lector = orden.ExecuteReader();
   if (lector.HasRows)
   {
       listmensaje = new ListMensaje();
       while (lector.Read())
            listmensaje.Add(FillDataRecord(lector));
   }
   lector.Close();
   conexion.Close();
   return listmensaje;
}
```

El método *HuerfanosById_ForoPaginados* sigue el mismo esquema que el anterior, pero recibe tres parámetros de tipo int (id, startIndexRows, maximumRows), que serán pasados como argumentos al procedimiento sprocMensajesHuerfanosById_ForoPaginados. El método *ListById_PadrePaginados*, es igual al descrito en este mismo párrafo pero utilizando el procedimiento sprocMensajesListById_PadrePaginados.

Método UltimaFechaById_Foro

A través de este método se averiguará cual fue la última fecha en la que se registró actividad en un determinado foro. Para ello se llama a la consulta correspondiente, pasándole como parámetro el identificador del foro en cuestión.

```
public static DateTime UltimaFechaById_Foro(int id)
{
  DateTime fecha = DateTime.MinValue;
  SqlConnection conexion = new SqlConnection(DALConfiguration.ConnectionString);
  SqlCommand orden = new SqlCommand("sprocMensajesUltimaFechaById_Foro",
conexion);
  orden.CommandType = CommandType.StoredProcedure;
  orden.Parameters.AddWithValue("@id_foro", id);
  conexion.Open();
  SqlDataReader lector = orden.ExecuteReader();
  if (lector.HasRows)
      while (lector.Read())
          fecha = lector.GetDateTime(1);
  lector.Close();
  conexion.Close();
  return fecha;
}
```

El método *UltimaFechaById_Padre*, es idéntico al anterior, pero el id aquí representa el Id del padre y no del foro, además el procedimiento utilizado es sprocMensajesUltimaFechaById_Padre.

UltimoUsuarioById_Padre, recibe el mismo argumento que los dos anteriores, pero retorna un string, por tanto defina una variable de este tipo en el método, el cual será valor de retorno, utilice el procedimiento sprocMensajesUltimoUsuarioById_Padre, y cambie en el ciclo de lectura la sentencia fecha = lector.GetDateTime(1) por user = lector.GetString(0, donde user es el valor de retorno.

Método NumById_Foro

```
public static int NumById_Foro(int id)
{
  int resultado = 0;
  SqlConnection conexion = new SqlConnection(DALConfiguration.ConnectionString);
  SqlCommand orden = new SqlCommand("sprocMensajesNumById_Foro", conexion);
  orden.CommandType = CommandType.StoredProcedure;
  orden.Parameters.AddWithValue("@id_foro", id);
  conexion.Open();
  SqlDataReader lector = orden.ExecuteReader();
  if (lector.HasRows)
      while (lector.Read())
          resultado = Convert.ToInt32((lector["nummensajes"].ToString()));
  lector.Close();
  conexion.Close();
  return resultado;
}
```

Utilice el procedimiento sprocMensajesNumById_Padre, para el método *NumById_Padre*, tome en cuenta el nombre del parámetro que recibe este procedimiento. Utilice el mismo esquema para el método *NumByUsuario* utilizando el procedimiento sprocMensajesNumByUsuario. Siga los mismos pasos para el método *NumPrincipalesById_Foro* utilizando el procedimiento sprocMensajesNumPrincipalesById_Foro.

4.5.5. Clase DALOpinión

Método NumOpinionesById_Noticia

```
public static int NumOpinionesById_Noticia(int Id)
{
   int resultado = 0;
   SqlConnection conexion = new SqlConnection(DALConfiguration.ConnectionString);
   SqlCommand orden = new SqlCommand("sprocOpinionesNumOpinionesById_Noticia",
conexion);
   orden.CommandType = CommandType.StoredProcedure;
   orden.Parameters.AddWithValue("@id_noticia", Id);
   conexion.Open();
   SqlDataReader lector = orden.ExecuteReader();
   if (lector.HasRows)
      while (lector.Read())
           resultado = Convert.ToInt32((lector["numopiniones"].ToString()));
   lector.Close();
   conexion.Close();
   return resultado;
}
```

Este método, es utilizado para obtener el número de opiniones vertidas sobre una determinada noticia, identificada por su Id.

4.5.6. Clase DALRespuestaEncuestaUsr

Método SelectNumVotosByIdRespuesta

Método utilizado para determinar el número de veces que los usuarios seleccionaron una determinada respuesta (identificada por su id) de una encuesta.

```
public static int SelectNumVotosByIdRespuesta(int id)
{
  int resultado = 0;
  SqlConnection conexion = new SqlConnection(DALConfiguration.ConnectionString);
  SqlCommand orden = new
SqlCommand("sprocRespuestasEncuestasUsrSelectNumVotosByIdRespuesta", conexion);
  orden.CommandType = CommandType.StoredProcedure;
  orden.Parameters.AddWithValue("@id_respuesta", id);
  conexion.Open();
  SqlDataReader lector = orden.ExecuteReader();
  if (lector.HasRows)
     while (lector.Read())
          resultado = Convert.ToInt32((lector["numvotos"].ToString()));
  lector.Close();
  conexion.Close();
  return resultado;
}
```

Método UsrHaVotadoById_Encuesta

Este método es utilizado para comprobar que un usuario, el pasado por argumento no ha votado en una determinada encuesta, identificada por el argumento id.

```
public static bool UsrHaVotadoById_Encuesta(int id, string usuario)
{
  bool resultado = false;
  SqlConnection conexion = new SqlConnection(DALConfiguration.ConnectionString);
  SqlCommand orden = new SqlCommand("sprocRespuestasEncuestasUsrHaVotado",
conexion);
  orden.CommandType = CommandType.StoredProcedure;
  orden.Parameters.AddWithValue("@usuario", usuario);
  orden.Parameters.AddWithValue("@id_encuesta", id);
  conexion.Open();
  SqlDataReader lector = orden.ExecuteReader();
  if (lector.HasRows)
      resultado = true;
  lector.Close();
  conexion.Close();
  return resultado;
}
```

5. CAPA DE LÓGICA DE NEGOCIO (BLL – Business Logic Layer)

5.1. Introducción

La capa de negocios o BLL (Layer Logic Business) es donde se comprueban las reglas de negocio y donde se manipulan los datos devueltos por la capa de acceso a datos antes de enviarlos a la capa de presentación.

Estaría formada por los objetos de negocio (BO, business object) y las clases encargadas de trabajar con dichos objetos (clases BLL), aunque éstos pueden ser considerados como una entidad independiente.

Cada clase BLL contiene los mismos métodos que su clase análoga en la capa inferior (clase DAL). Estos métodos realizarán funciones de validación y de filtrado de datos antes de llamar a sus métodos parejos en la capa de acceso a datos.

Para una mayor organización estas clases serán agrupadas en un mismo espacio de nombres (*PortalWeb.BLL*).

Para añadir una clase BLL al Portal realice los siguientes pasos:

- Seleccionar App_Code (creada anteriormente)
- Ahora cree una carpeta dentro de App_Code destinada al almacenamiento de las clases BLL, llamada *Business Logia.*
- Clic derecho sobre la carpeta *Business Logia.*
- Agregar nuevo elemento ..
- Seleccione la plantilla **Clase**
- Asígnele nombre a la clase (*BLLNoticia*)

5.2. Diagrama de clases de la capa BLL

La **Fig. 57** muestra el diagrama de clases de la capa BLL, por tanto deberá crear cada una de las clases que se muestran en el diagrama. Para crear este diagrama una vez añadida todas las

clases necesarias, deberá seguir los mismo pasos que en el creación del diagrama *Business Object* y añadir desde le *Explorador de soluciones* todas las clases de la carpeta *Business Logic*.

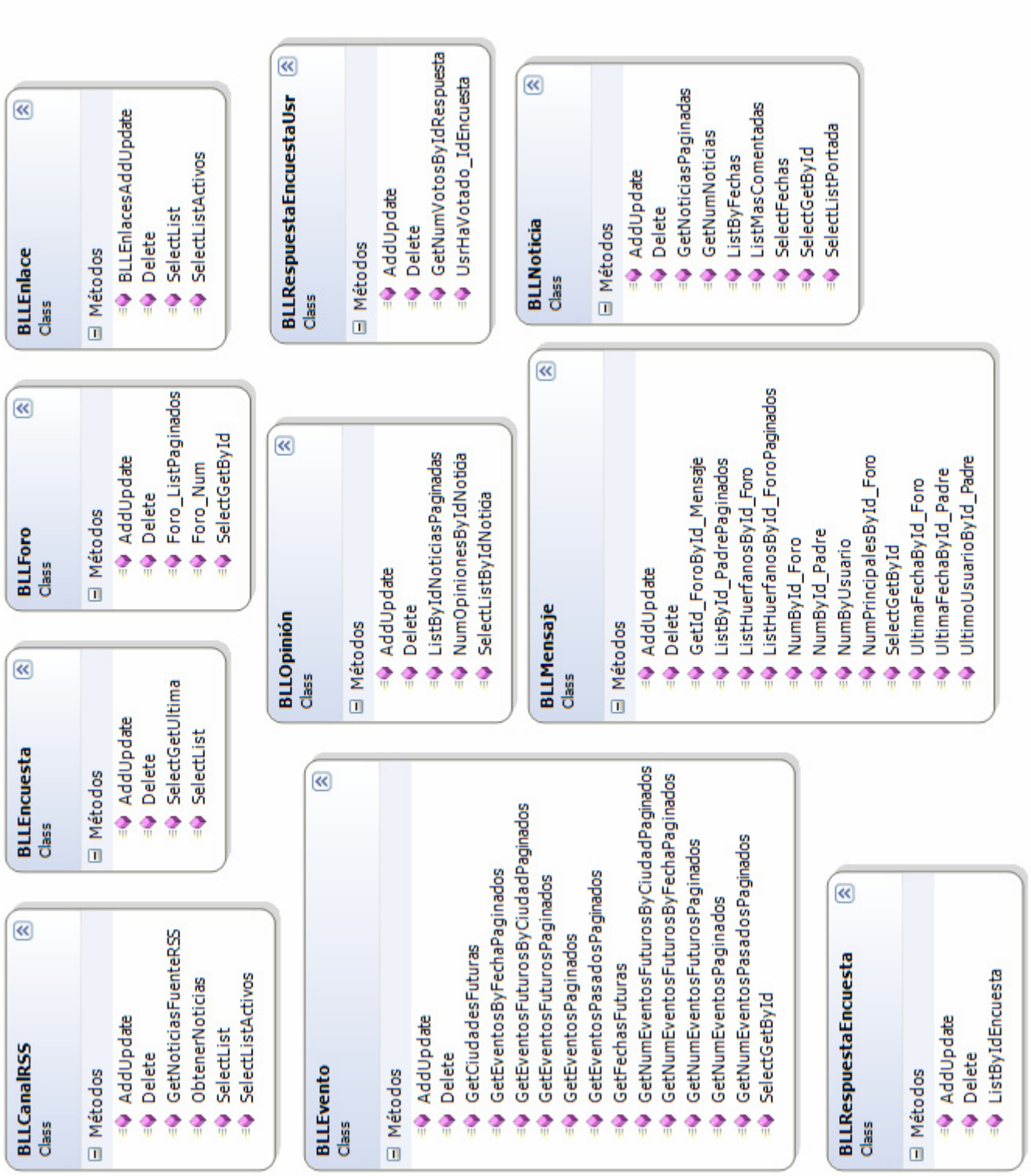


Fig. 57 Diagrama de clases de la capa BLL.

Cada unas de estas clases deberá tratar con objetos de negocio, y también con la capa de Acceso a Datos, para poder acceder a los objetos de negocio, se deberá hacer visible el espacio de nombre *PortalWeb.BO* con la cláusula *using*, esto en cada una de las clases y también se debe hacer visible el espacio *PortalWeb.DAL*:

```
using PortalWeb.BO;
using PortalWeb.DAL;
```

5.3. Implementación de las clases de la capa BLL

Para poder vincular las diferentes clases BLL de la capa de lógica de negocio con un objeto *ObjectDataSource*, se debe añadir la siguiente sentencia antes de la declaración de la clase.

```
[DataObjectAttribute()]
public class BLLNoticia
{
}
```

De esta forma se inicializa una nueva instancia de la clase *DataObjectAttribute*. Esta clase identifica un tipo como objeto adecuado para enlazarlo a un objeto *ObjectDataSource* y se encuentra dentro del espacio de nombre *System.ComponentModel*.

Se hará uso de la clase *DataObjectMethodAttribute* que sirven para identificar un método de operación de datos expuesto por un tipo, el tipo de operación que realiza el método y si es el método de datos predeterminado. *DataObjectMethodAttribute* se puede utilizar para identificar los métodos de operación de datos en un tipo marcado con el atributo *DataObjectAttribute*, de modo que los llamadores pueden identificarlos más fácilmente mediante la reflexión. Cuando el atributo *DataObjectMethodAttribute* se aplica a un método, describe el tipo de operación que realiza el método e indica si dicho método es el método de operación de datos predeterminado de un tipo. Los componentes como el control *ObjectDataSource* y la clase *ObjectDataSourceDesigner* examinan los valores de este atributo, cuando está presente, para determinar el método de datos al que llamar en tiempo de ejecución.

5.4. Clase BLLNoticia

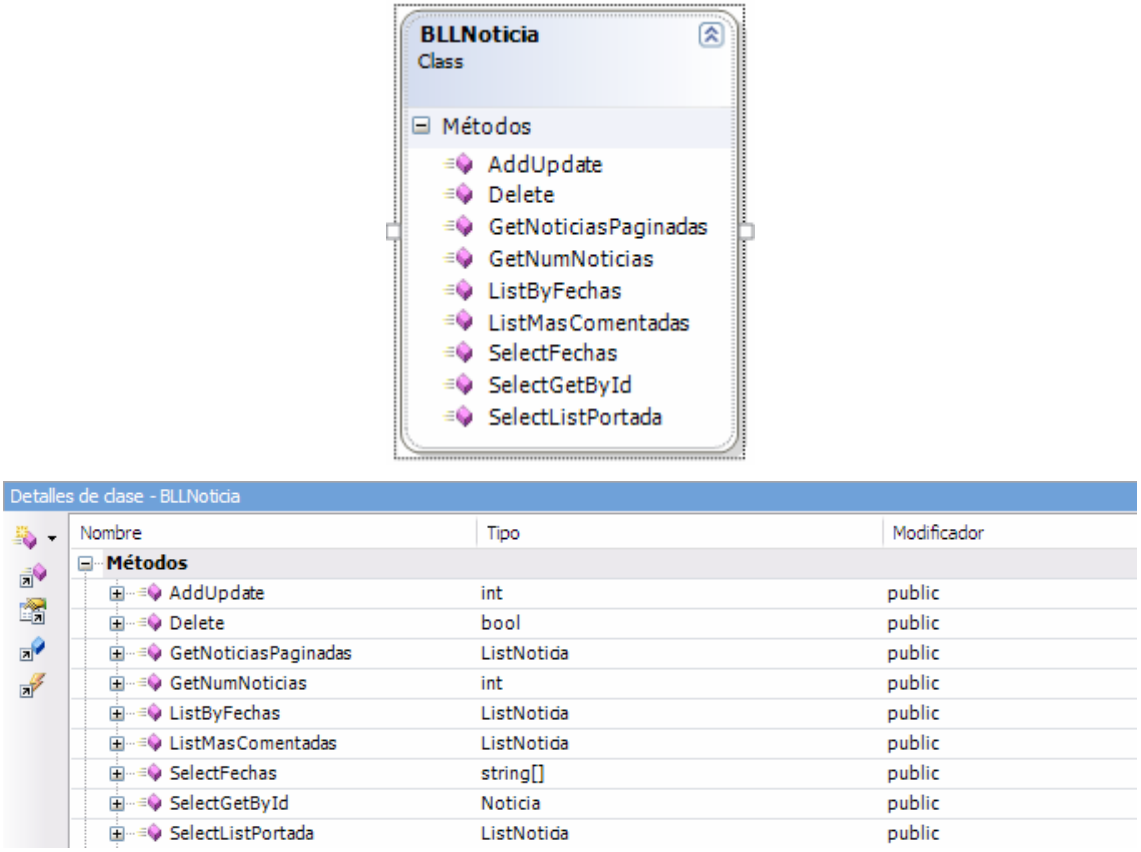


Fig. 58 Clase BLLNoticia.

Las figuras anteriores muestran los detalles de la clase *BLLNoticia*, como se observa en la **Fig. 58** aparece el nombre del método. La parte inferior es simplemente una ampliación de la dicha figura observe que todos los métodos son públicos.

Una vez creada la clase BLLNoticia, proceda a definir los métodos mostrados correspondientes.

5.4.1. Método Delete

```
[DataObjectMethod(DataObjectMethodType.Delete, true)]
public  bool Delete(Noticia  miNoticia)
{
   return DALNoticia.Delete(miNoticia._Id_Noticia);
}
```

El método Delete recibe como parámetro un objeto de negocio de tipo Noticia y llama al método Delete de la capa de acceso a datos pasándole como parámetro el identificador de dicho objeto. Devuelve un booleano indicando si se ha realizado satisfactoriamente o no la operación. Como se observa este será el método de datos predeterminado para la operación Delete que expondrá el objeto de datos.

5.4.2. Método SelectGetById

```
[DataObjectMethod(DataObjectMethodType.Select, true)]
public static Noticia  SelectGetById(int Id)
{
    return DALNoticia.SelectGetById(Id);
}
```

El método SelectGetById devuelve la noticia correspondiente a un determinado identificador recibido como parámetro. Al igual que en los casos anteriores haremos uso del método homólogo de la capa de acceso a datos.

5.4.3. Método SelectListPortada

```
[DataObjectMethod(DataObjectMethodType.Select, true)]
public static ListNoticia SelectListPortada()
{
    return DALNoticia.SelectListPortada();
}
```

Este método devuelve una colección de objetos de tipo Noticia, para lo que llama al método SelectLsitPortada de la capa de acceso a datos. Dicha colección será la que aparezca en la página principal del sitio Web. Observe que representa una operación Select.

5.4.4. Método ListByFecha

```
[DataObjectMethod(DataObjectMethodType.Select , true)]
public  ListNoticia ListByFechas(string fecha)
{
    return DALNoticia.ListByFechas(Convert.ToDateTime(fecha));
}
```

Este método devuelve una colección de noticias publicadas en una determinada fecha. Recibe como parámetro la fecha en formato String y llama al método DALNoticia.ListByFechas pasándole como parámetro dicha fecha convertida al formato DateTime.

5.4.5. Método ListMasComentadas

```
[DataObjectMethod(DataObjectMethodType.Select, true)]
public    ListNoticia  ListMasComentadas()
{
    return DALNoticia.ListMasComentadas();
}
```

Este método llama al método ListMasComentadas de la capa de acceso a datos para obtener una colección de las noticias con mayor número de opiniones de la base de datos.

5.4.6. Método SelectFechas

Con este método obtenemos un array del tipo String con todas las fechas en las que se ha publicado alguna noticia.

```
[DataObjectMethod(DataObjectMethodType.Select, true)]
public static string[] SelectFechas()
{
     return DALNoticia.SelectFechas();
}
```

5.4.7. Método AddUpdate

```
[DataObjectMethod(DataObjectMethodType.Update, true)]
public static int AddUpdate(Noticia  miNoticia)
{
    using (TransactionScope myTransactionScope = new TransactionScope())
    {
          int idOpinión = DALNoticia.AddUpdate(miNoticia);
          foreach (Opinión opinión in miNoticia._Opiniones)
         {
              opinión._Id_Noticia = idOpinión;
              DALOpinión.AddUpdate(opinión);
          }
```

```
            miNoticia._Id_Noticia = idOpinión;
            myTransactionScope.Complete();
            return idOpinión;
        }
    }
```

El método AddUpdate tiene como objetivo principal llamar al método AddUpdate de la capa de acceso a datos para llevar a cabo una inserción o una actualización de una noticia dada. Recibe como parámetro un objeto de negocio Noticia, y devuelve el identificador del mismo.

Note que por cada opinión que tenga asociada la noticia recibida como parámetro (minoticia._Opiniones), se llama al método AddUpdate de la clase DALOpinión, y se le asocia el identificador de la noticia que ha sido añadida o actualizada. Observe que todas estas operaciones están dentro de un bloque transaccional using (TransactionScope myTransactionScope = new TransactionScope()) y que finalice con la sentencia myTransactionScope.Complete(), esto para garantizar la consistencia de los datos, para ello es necesario hacer uso del espacio de nombres *using System.Transactions*.

5.4.8. Método GetNoticiasPaginadas

```
[DataObjectMethod(DataObjectMethodType.Select, true)]
public  ListNoticia GetNoticiasPaginadas(int startRowIndex, int maximumRows)
{
    return DALNoticia.GetNoticiasPaginadas(startRowIndex, maximumRows);
}
```

Observe que este método llama a su homólogo en la capa de acceso de datos, pasándole los valores que él recibe como parámetros, los que representan el rango de la selección de las noticias.

5.4.9. Método GetNumNoticias

```
[DataObjectMethod(DataObjectMethodType.Select, true)]
public int GetNumNoticias()
{
    return DALNoticia.GetNumNoticias();
}
```

5.5. Peculiaridades del resto de las clases BLL.

Define los métodos se muestran en la **Fig. 57** para cada una de las clases BLL, a continuación se muestran los procedimientos que difieren de los ejemplares mostrados. Recuerde como forma genérica, especificar el tipo de operación que representa el método (Delete para todos los métodos de igual nombre, Update para los métodos AddUpdate y Select para el resto), llame al método de la clase DAL correspondiente, pase los argumentos necesarios y retorne el valor adecuado.

5.5.1. Clase BLLEncuesta

Método AddUpdate

```
[DataObjectMethod(DataObjectMethodType.Update, true)]
public static int AddUpdate(Encuesta miencuesta)
{
   using (TransactionScope myTransactionScope = new TransactionScope())
   {
       int idEncuesta = DALEncuesta.AddUpdate(miencuesta);
       foreach (RespuestaEncuesta respuesta in miencuesta._Respuestas)
       {
           respuesta._Id_Encuesta = idEncuesta;
           DALRespuestaEncuesta.AddUpdate(respuesta);
           foreach (RespuestaEncuestaUsr respuestausuario in respuesta._RespuestasUsr)
           {
                respuestausuario._Id_Encuesta = idEncuesta;
                DALRespuestaEncuestaUsr.AddUpdate(respuestausuario);
            }
       }
       miencuesta._Id_Encuesta = idEncuesta;
       myTransactionScope.Complete();
       return idEncuesta;
   }
}
```

Como se puede observar, aquí se actualiza o inserta una Encuesta, las Respuestas a dicha Encuesta y las Respuestas seleccionada por los usuarios, en el método AddUpdate de la clase BLLRespuestaEncuesta, además de llamar al método DALRespuestaEncuesta.AddUpdate, deberá llamara al método AddUpdate de la clase DALRespuestaEncuestaUsr, proceda de forma similar en la método AddUpdate de la clase BLLForo, llamando también al método

AddUpdate de la clase BLLMensaje, también deberá actuar de forma similar la clase BLLMensaje.

En el método SelectGetUltima, deberá llamar a su homólogo en la capa DAL, pasándole como parámetro una variable local de tipo DataTime, inicializada con la propiedad Today de dicha clase.

5.5.2. Clase BLLCanalRSS

Método GetNoticiasFuenteRSS

Este método recibe como parámetro un objeto de tipo CanalRSS y retorna un objeto de tipo FuenteRss. En primer lugar cree un objeto FuenteRss cuyo titulo será el del objeto CanalRSS recibido. A continuación cree un objeto XmlTextReader (rssReader) y una variable de tipo entero cuya misión será el recuento de elementos leídos dentro de un determinado canal.

Seguidamente recupere la fuente Rss mediante la clase WebRequest y cargue los datos en el lector rssReader. A continuación se crea un bucle While para leer el archivo Xml, que finalizará cuando la lectura del archivo Xml haya llegado a su fin o el contador de elementos haya alcanzado su valor límite, que en nuestro caso será de cinco.

Dentro del bucle se buscan los elementos del archivo Xml que correspondan con el inicio de una fuente rss de los diferentes tipos existentes. Una vez obtenido el tipo de fuente rss que se está leyendo, mediante la sentencia switch se irá comparando dicho tipo con cada uno de los posibles para ejecutar en cada caso el código pertinente. Dentro de cada uno de los “case” e hará prácticamente lo mismo. Primeramente se buscan los elementos del archivo Xml que correspondan con los ítems del canal, y se crea un nuevo lector del tipo XmlReader (itemReader) para leer la información de los mismos. A continuación se crea un nuevo objeto ItemFuenteRSS para almacenar los datos del elemento. Acto seguido le asignamos el valor devuelto por el método ReadSubTree del Xml principal al último lector creado, que abarcará los datos actuales del elemento Rss.

El código llama al método Read del mismo para encontrar cada uno de los elementos de datos de interés. Cuando los encuentra se almacena el valor de los datos en el campo apropiado del objeto ItemFuenteRSS. Al final de este bucle añadimos dicho objeto a la colección de Ítems del objeto FuenteRSS y aumentamos en una unidad el contador de elementos.Cabe resaltar que para cada tipo de rss, el segundo lector (itemReader) buscará unos u otros elementos en el archivo dependiendo de la naturaleza de la fuente.

```
public static FuenteRSS GetNoticiasFuenteRSS(CanalRSS canalrss)
{
   FuenteRSS frss = new FuenteRSS();
   frss._title  = canalrss._Título ;
   XmlTextReader rssReader = null;
   int itemCount = 0;
   try
   {
      WebRequest rssFeed = WebRequest.Create(canalrss._Url);
      rssReader = new XmlTextReader(rssFeed.GetResponse().GetResponseStream());
      while (rssReader.Read() && itemCount<5) //Esta leyendo el archivo xml
      {
         if ((rssReader.IsStartElement()) && ("rss" == rssReader.LocalName))
         {
            if ((rssReader.GetAttribute("version")) == "2.0")
                  frss._tiporss = TipoRSS.RSS20;
            else
                  frss._tiporss= TipoRSS.RSS091;
         }

         else if ((rssReader.IsStartElement()) && ("rdf:RDF" == rssReader.Name))
            frss._tiporss = TipoRSS.RSS10;

         else if ((rssReader.IsStartElement()) && ("feed" == rssReader.LocalName))
            frss._tiporss= TipoRSS.Atom;

         switch (frss._tiporss )
         {
            case TipoRSS.RSS091:
                  if ((rssReader.IsStartElement()) && ("item"== rssReader.LocalName))
                  {
                     XmlReader itemReader = null;
                     try
                     {
                        ItemFuenteRSS nuevoitem = new ItemFuenteRSS();
                        itemReader = rssReader.ReadSubtree();
                        while (itemReader.Read())
                        {
                           if (itemReader.IsStartElement())
                           {
                              if ("title" == itemReader.LocalName)
                                 nuevoitem._title = itemReader.ReadString();
                              else if ("description" == itemReader.LocalName)
                              {
```

```
                        string newDescription = itemReader.ReadString();
                        if (newDescription.Length > 200)
                        newDescription = newDescription.Substring(0,200) + " ...";
                        nuevoitem._description  = newDescription;
                    }
                    else if ("link" == itemReader.LocalName)
                        nuevoitem._link = itemReader.ReadString();
                    else if ("pubDate" == itemReader.LocalName)
                        nuevoitem.PubDate = itemReader.ReadString();
                }
            }
            frss._items.Add(nuevoitem);
            itemCount++;
        }
        finally
        {
            if (itemReader != null)
                itemReader.Close();
        }
    }
break;

  case TipoRSS.Atom:
    if ((rssReader.IsStartElement()) && ("entry"==rssReader.LocalName))
    {
        XmlReader itemReader = null;
        try
        {
            ItemFuenteRSS nuevoitem = new ItemFuenteRSS();
            itemReader = rssReader.ReadSubtree();
            while (itemReader.Read())
            {
                if (itemReader.IsStartElement())
                {

                    if ("title" == itemReader.LocalName)
                        nuevoitem._title = itemReader.ReadString();
                    else if ("link" == itemReader.LocalName)
                        nuevoitem._link = itemReader["href"].ToString();
                    else if ("created" == itemReader.LocalName)
                        nuevoitem.PubDate = itemReader.ReadString();
                    else if ("summary" == itemReader.LocalName)
                    {
                        string newDescription = itemReader.ReadString();
                        if (newDescription.Length > 200)
                            newDescription = newDescription.Substring(0, 200) + " ...";
                        nuevoitem._description = newDescription;
                    }
                }
            }
            frss._items.Add(nuevoitem);
            itemCount++;
        }
        finally
        {
            if (itemReader != null)
```

```
                    itemReader.Close();
                }
            }
            break;

            case TipoRSS.RSS10:
                if ((rssReader.IsStartElement()) && ("item" == rssReader.LocalName))
                {
                    XmlReader itemReader = null;
                    try
                    {
                        ItemFuenteRSS nuevoitem = new ItemFuenteRSS();
                        itemReader = rssReader.ReadSubtree();
                        while (itemReader.Read())
                        {
                            if (itemReader.IsStartElement())
                            {
                                if ("title" == itemReader.LocalName)
                                    nuevoitem._title = itemReader.ReadString();
                                else if ("description" == itemReader.LocalName)
                                {
                                    string newDescription = itemReader.ReadString();
                                    if (newDescription.Length > 200)
                                        newDescription = newDescription.Substring(0,200) + " ...";
                                    nuevoitem._description = newDescription;
                                }
                                else if ("link" == itemReader.LocalName)
                                    nuevoitem._link = itemReader.ReadString();
                                else if ("dc:date" == itemReader.Name)
                                    nuevoitem.PubDate = itemReader.ReadString();
                            }
                        }
                        frss._items.Add(nuevoitem);
                        itemCount++;
                    }
                    finally
                    {
                        if (itemReader != null) itemReader.Close();
                    }
                }
                break;

                case TipoRSS.RSS20:
                    if ((rssReader.IsStartElement()) && ("item" ==rssReader.LocalName))
                    {
                        XmlReader itemReader = null;
                        try
                        {
                            ItemFuenteRSS nuevoitem = new ItemFuenteRSS();
                            itemReader = rssReader.ReadSubtree();
                            while (itemReader.Read())
                            {
                                if (itemReader.IsStartElement())
                                {
                                if ("title" == itemReader.LocalName)
                                    nuevoitem._title = itemReader.ReadString();
```

```
                        else if ("description" == itemReader.LocalName)
                        {
                            string newDescription = itemReader.ReadString();
                            nuevoitem._description = newDescription;
                        }
                        else if ("link" == itemReader.LocalName)
                            nuevoitem._link= itemReader.ReadString();
                        else if ("pubDate" == itemReader.LocalName)
                            nuevoitem.PubDate = itemReader.ReadString();
                        }
                    }
                    frss._items.Add(nuevoitem);
                    itemCount++;
                }
                finally
                {
                    if (itemReader != null) itemReader.Close();
                }
            }
            break;
        } //Fin Swicth
    }//FIn WHILE
}//FIN TRY
catch (System.UriFormatException)
 {
       // Invalid URI for source RSS feed: continue processing
  }
 finally
{
    if (rssReader != null) rssReader.Close();
}
return frss;
}
```

Método ObtenerNoticias

Este método devuelve un listado de objetos FuenteRSS. En cada uno de estos objetos se recogerán todas las propiedades de cada canal proporcionadas por la fuente. Además en la propiedad Ítems de cada objeto almacenaremos toda la información referente a los ítems ofrecidos por el canal.

Para ello, por cada objeto CanalRSS obtenido de la llamada al método SelectListActivos se crea un objeto CanalRSS. Dicho objeto será rellenado con la información devuelta por la llamada GetNoticiasFuenteRSS(). Este método recibe como parámetro un objeto de tipo CanalRSS y retorna un objeto de tipo FuenteRSS.

```
[DataObjectMethod(DataObjectMethodType.Select, false)]
        public static ListFuenteRSS ObtenerNoticias()
        {
        //obtengo de la base de datos las fuentes activas
            ListCanalRSS  listacanal =   SelectListActivos();
            ListFuenteRSS listafuentes = new ListFuenteRSS();
            foreach (CanalRSS  ri in listacanal)
            {
                //Por cada fuente, descargo las noticias y las devuelvo en una coleccion
                FuenteRSS frss;
                frss = GetNoticiasFuenteRSS(ri);
                //cambio la descripcion de la fuente por la que he puesto en la base de datos
                frss._description = ri._Descripción;
                listafuentes.Add(frss);
            }
            return listafuentes;
        }
```

6. CAPA DE PRESENTACIÓN

6.1. Introducción

La capa de presentación es la más elevada de las que componen la estructura de 3 capas de la aplicación. Bajo ella subyacen la de lógica de negocio y la de acceso a datos. Constituye la parte visible de cara al usuario ya que será la parte con la cual se verá obligado a interactuar.

La capa de presentación se compone de varias páginas Web (.aspx), algunas de ellas formadas a su vez por diferentes controles de usuario (.ascx).

Todas las páginas que componen el sitio Web contendrán una serie de controles comunes. Por ello, será necesaria la creación de una página maestra donde situaremos dichos controles, los que serán heredados por el resto de páginas.

Su diseño se ajustará a un esquema básico de tres columnas. Las dos laterales, de menor anchura, estarán destinadas albergar los controles comunes y la central será la encargada de mostrar los contenidos propios de cada página.

Los controles comunes y visibles en todas las páginas serán:

- Login: para el registro e inicio de sesión de los usuarios.
- Buscador: permite realizar búsquedas en Google desde el sitio Web.
- Noticias más comentadas: muestra las noticias que han sido objeto de un mayor número de opiniones por parte de los usuarios.
- Calendario de eventos: muestra los acontecimientos a efectuar en fechas venideras.
- Encuesta: muestra la última encuesta publicada para poder así participar en ella.

Para la ubicación de estos controles, se hará uso de paneles, a los que se les aplicará la técnica de las esquinas redondeadas, para hacer así más atractiva su apariencia.

A su vez, se dispone de una cabecera en la parte superior para dar cabida, principalmente, al logo del Portal así como a un menú de navegación.

Para el desarrollo general del portal, se ha recurrido al uso de hojas de estilo CSS como patrón para definir su aspecto. De esta forma resulta muy sencillo cambiar su aspecto.

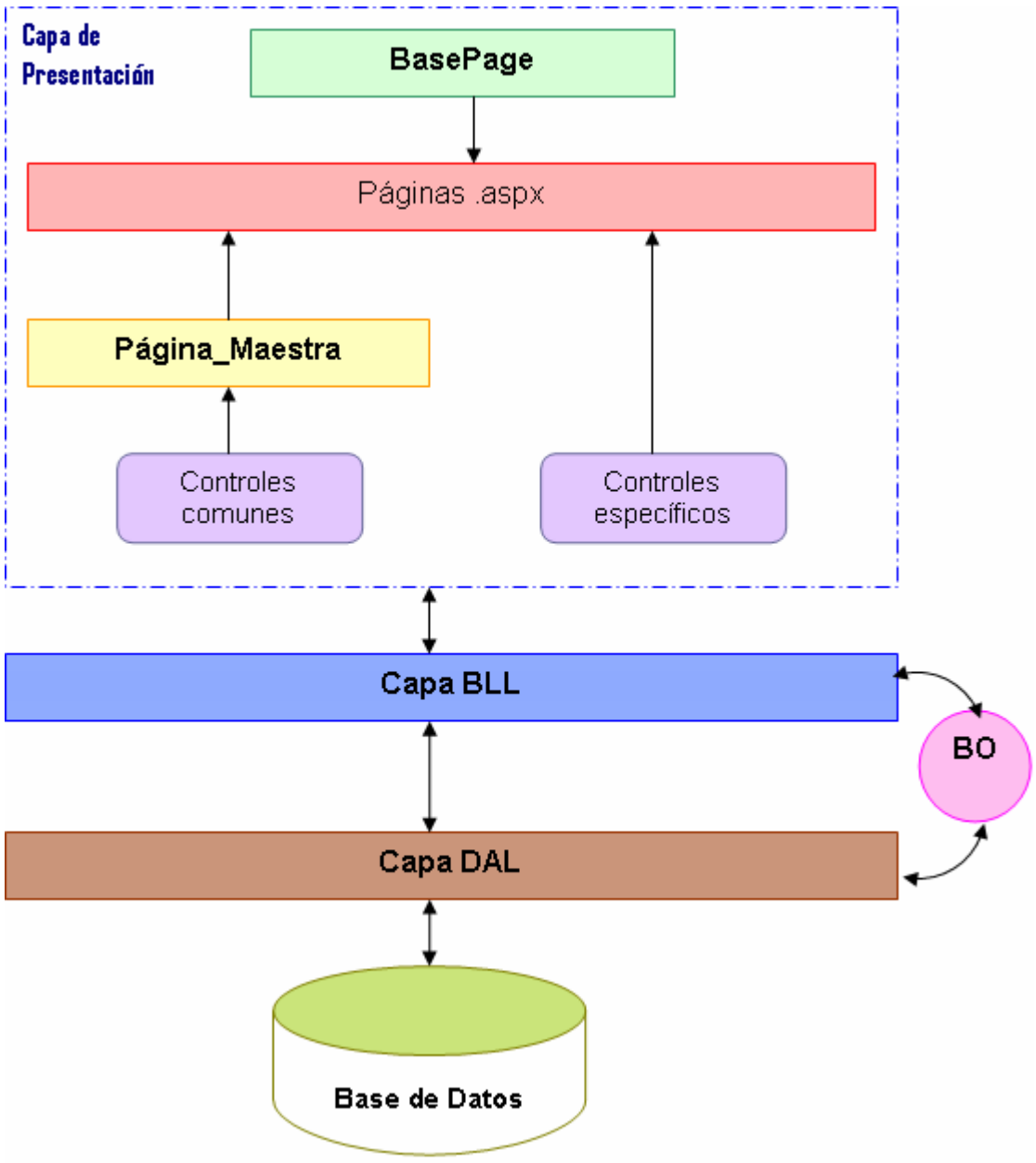


Fig. 59 Arquitectura del Portal de Noticias.

Los contenidos y funciones disponibles en el sitio Web no serán los mismos para todos los usuarios, pues se ha optado por definir cuatro tipos de roles diferentes para regular su uso y acceso. Cada uno de ellos se ajustará a las necesidades de cada tipo de usuario y favorecerá un correcto manejo y administración del portal. Los roles serán cuatro: usuarios anónimos, para

visitantes no registrados; usuarios registrados, para usuarios previamente registrados; moderador, para supervisar el funcionamiento del foro; y administrador, para aquellas personas encargadas de gestionar y mantener el sitio Web.

Cabe también destacar el uso masivo del control ObjectDataSource para la vinculación de los diferentes controles de usuario con la capa BLL. Para mostrar los contenidos relativos a las diferentes secciones existentes en el sitio, se hace uso de diversos tipos de controles, entre los que se encuentran los del tipo Gridview, o Datalist.

6.2. Controles comunes

6.2.1. Control Menú Principal

6.2.1.1. Descripción

Uno de los elementos fundamentales del sitio Web es sin duda el menú principal. Este sirve para acudir a una u otra sección del portal con tan sólo hacer clic en la pestaña indicada. Se trata pues de un menú de navegación por pestañas, ubicado en la cabecera de la página principal, y que por tanto será heredada por el resto.

Las pestañas aparecerán de un determinado color por defecto, y al situar el cursor sobre ellas, su apariencia cambiará. Asimismo, dependiendo del perfil seleccionado por cada usuario, el aspecto de las pestañas cambiará favoreciendo así la personalización del portal por los navegantes, todos estos aspectos serán tratados más adelante.

6.2.1.2. Desarrollo

Para elaborar el menú, nos fundamentaremos en una lista de objetos definidos por una serie de etiquetas, cuyo cuerpo consiste en una referencia a una página de destino a la que acudiremos tras hacer clic en el botón correspondiente. Todo ello conformará el texto de un control de tipo Literal, al que tras aplicar una hoja de estilos css dará lugar a las pestañas del menú principal. Para ir organizando adecuadamente el directorio del sitio Web realice los siguientes pasos:

- Añada una carpeta denominada Controles en la raíz del sitio (http://localhost/PortalWeb/). Albergarán la mayoría de los controles de usuario.
- Agregue sobre la carpeta Controles un Nuevo Elemento.
- Seleccione como plantilla *Control se usuario Web.*
- Asígnele como nombre *Menú_Principal*
- Clic en Agregar
- Desde la vista de Diseño, añada un control de tipo Literal (Literal1)

A continuación estando en la vista Código (Archivo Menú_Principal.ascx.cs), edite el controlador del evento Load que será el encargado de general el código html que formará el menú.

Para crear listados de objetos en html, primero hay que definir el tipo de lista y después deben insertarse cada uno de los elementos que la forman. Para ello, emplee la etiqueta <ul>, que servirá para declarar una lista de objetos no ordenada. Una vez abierta la lista debe insertar, como se mencionó anteriormente, cada uno de los elementos que la forman. Para ello utilice una nueva etiqueta: <li>. Situé una al inicio y otra al final del contenido del elemento, que en este caso será una referencia a la página a la que se desee dirigirse.

Para esta parte, añada a la cadena de caracteres la etiqueta <a href, acompañada de la cadena que contiene la dirección de la página de destino. A continuación incluya el nombre de la etiqueta que se mostrará, situada entre dos etiquetas , etiqueta para marcar bloques de texto. Para concluir, cierre la etiqueta , y acto seguido también a etiqueta .

Actúe de la misma forma con todos y cada uno de los elementos de la lista, y acto seguido, la cierre con otra etiqueta .

El conjunto de toda esta cadena dará lugar al texto de un elemento de tipo literal, que conformará en sí mismo el menú de pestañas. El control Literal se utiliza principalmente cuando el contenido se agrega a la página de forma dinámica, es decir, este no agrega ningún elemento HTML al texto. De esta manera se da forma al menú de pestañas dinámicamente. El código completo se muestra a continuación:

```
protected void Page_Load(object sender, EventArgs e)
{
        string str = "";
        str = ("<ul>");//Lista de objetos no ordenada
        str += ("<li>"); //Insertar los elementos en la lista
        string path = Request.ApplicationPath;
        if (path == "/")
            path = "";
        str += ("<a href='" + path +"/Controles/NoticiasRSS/NoticiasRss.aspx'><span>Noticias
vía RSS</span></a>");
        str += ("</li>");
        str += ("<li>");
        str += ("<a href='" + path +
"/Controles/Foro/IndiceForos.aspx'><span>Foros</span></a>");
        str += ("</li>");
        str += ("<li>");
        str += ("<a href='" + path +
"/Controles/Enlaces/Enlaces.aspx'><span>Enlaces</span></a>");
        str += ("</li>");
        str += ("<li>");
        str += ("<a href='" + path +
"/Controles/Noticias/BuscaNoticias.aspx'><span>Noticias</span></a>");
        str += ("</li>");
        str += ("<li>");
        str += ("<a href='" + path + "/Default.aspx'><span>Portada</span></a>");
        str += ("</li>");
        str += ("</ul>");
       Literal1.Text=str;
}
```

6.2.1.3. Estilo CSS

Ahora hay que proceder a definir los estilos que van a afectar al Menú Principal. Para ello y adelantándonos un poco en la aplicación de Temas, realice las siguientes opciones:

- Clic derecho sobre el directorio del sitio (*http://localhost/PortalWeb*)
- Agregar Carpeta ASP .NET
- Seleccionar Tema
- Asignarle un nombre (Azul)

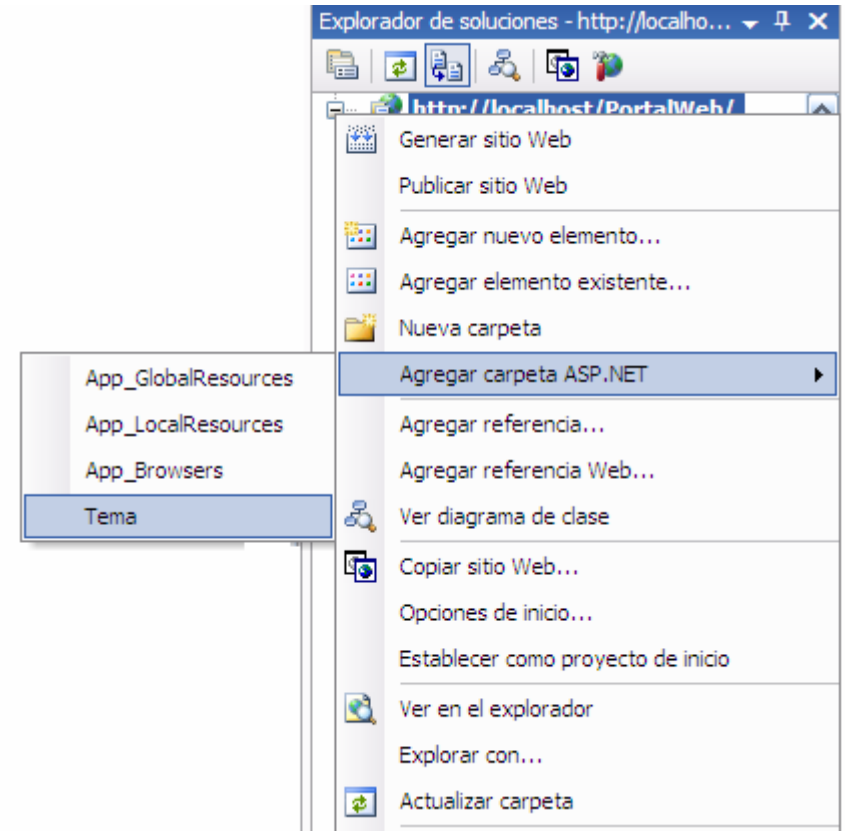


Fig. 60 Crear la carpeta Tema.

A continuación cree un nuevo archivo de tipo Hoja de estilo en la carpeta creada anteriormente y denomínelo Hoja_Estilo.

Es hora de definir los estilos que se aplicarán al menú creado anteriormente, por tanto edite el archivo Hoja_Estilo de la siguiente manera:

```
/*Estilo del Menú Principal,División*/
#MenuPrincipal
{
float: right; /*Alinear el elementos a la derecha*/
width: 100%;
background: transparent;
font-size: 93%;
line-height:normal;
border-bottom: 1px solid Gray ;
font: bold 11px/1.5em Verdana;
text-align: center;
}

/*Estilo de la lista del Menú Principal*/
#MenuPrincipal ul
```

```
{
margin: 0;
padding: 10px 10px 0 50px;
list-style: none;
}

/*Estilo de cada uno de los elementos de la lista*/
#MenuPrincipal li
{
display: inline;
margin: 0;
padding: 0;
}
#MenuPrincipal a
{
float: right;
background: url( "Imagenes/Menu2.PNG" ) no-repeat left top;
margin: 1;
padding: 4 4 4 4px;
text-decoration: none;
}
#MenuPrincipal a span
{
float: right;
display: block;
background: url( "Imagenes/Menu1.jpg" ) no-repeat right top;
padding: 5px 15px 4px 6px;
color: Black;  /*Color del texto*/
}
#MenuPrincipal a span
{
float: none;
}
#MenuPrincipal a:hover span
{
color:#FFF;   /*Color cuando se pone encima el cursor*/
}
#MenuPrincipal a:hover
{
background-position: 0% -42px;
}
#MenuPrincipal a:hover span
{
background-position: 100% -42px;
}
#MenuPrincipal #current a
{
background-position: 0% -42px;
}
#MenuPrincipal #current a span
{
background-position: 100% -42px;
color: #FFF;
}
```

Esta clase afectará al conjunto del menú principal, y en ella se definen algunos aspectos, tales que la imagen se anclará al margen derecho, que el texto estará alineado al centro, que el color de fondo será transparente entre otras características.

6.2.2. Control Menú Administrador

6.2.2.1. Descripción

Para facilitar en la medida de lo posible las tareas de gestión del portal por parte del administrador, se ha incluido un menú que permitirá el acceso a los diferentes modos de gestión y que sólo estará disponible para quien pertenezca a dicho rol de administrador.

Se trata de un menú de pestañas ubicado en el margen izquierdo de la página maestra y que por tanto será heredado por el resto. Las pestañas aparecerán de un determinado color por defecto, y al situar el cursor sobre ellas, su apariencia cambiará.

Asimismo, dependiendo del perfil seleccionado por cada usuario, el aspecto de las pestañas cambiará favoreciendo así la personalización del portal.

Fig. 61 Control Menú Administrador.

6.2.2.2. Desarrollo

Para elaborar el menú, se utilizará una lista de objetos definidos por una serie de etiquetas, cuyo cuerpo consiste en una referencia a una página de destino a la que se acudirá tras hacer clic en el botón correspondiente. Todo ello conformará el texto de un control de tipo Literal, al que tras aplicar una hoja de estilos CSS dará lugar a los elementos del menú admin.

Para ello siga los siguientes pasos:

- Dentro de la carpeta Controles añada una nueva denominada Admin. (Visible únicamente para los usuarios administradores).
- Agregue sobre la carpeta Admin un Nuevo Elemento.
- Seleccione como plantilla *Control se usuario Web.*
- Asígnele como nombre *Menú_Admin*
- Clic en Agregar
- Desde la vista de Diseño, añada un control de tipo Literal (Literal1)

A continuación estando en la vista Código (Archivo Menú_Admin.ascx.cs), edite el controlador del evento Load que será el encargado de general el código html que formará el menú.

```
protected void Page_Load(object sender, EventArgs e)
   {
      int idadm = 0;
      int.TryParse(Request["idadm"], out idadm);
      string str = "";
      str = ("<ul>");
      str += ("<li>");
      string path = Request.ApplicationPath;
      if (path == "/")
         path = "";
      str += ("<a " + EscribeA(1,idadm) + "href='" + path +
      "/Editores/GestionEncuestas.aspx?idadm=1'><span>Gestión Encuestas
</span></a>");
      str += ("</li>");
      str += ("<li>");
      str += ("<a " + EscribeA(2,idadm) + "href='" + path +
      "/Editores/GestionEnlaces.aspx?idadm=2'><span>Gestión Enlaces </span></a>");
      str += ("</li>");
      str += ("<li>");
      str += ("<a " + EscribeA(3,idadm) + " href='" + path +
      "/Editores/GestionEventos.aspx?idadm=3'><span>Gestión Eventos </span></a>");
      str += ("</li>");
      str += ("<li>");
      str += ("<a " + EscribeA(4, idadm) + " href='" + path +
      "/Editores/GestionFuentesRSS.aspx?idadm=4'><span>Gestión Fuentes RSS
</span></a>");
      str += ("</li>");
      str += ("<li>");
      str += ("<a " + EscribeA(5,idadm) + " href='" + path +
      "/Editores/GestionForos.aspx?idadm=5'><span>Gestión Foros </span></a>");
      str += ("</li>");
      str += ("<li>");
      str += ("<a " + EscribeA(6, idadm) + " href='" + path +
      "/Editores/GestionNoticias.aspx?idadm=6'><span>Gestión Noticias</span></a>");
      str += ("</li>");
      str += ("<li>");
```

```
        str += ("<a " + EscribeA(7, idadm) + " href='" + path +
        "/Editores/Membership.aspx?idadm=7'><span>Gestión Usuarios </span></a>");
        str += ("</li>");
        str += ("</ul>");
        Literal1.Text = str;
    }
```

En primer lugar, en el evento Page_Load se declara una variable de tipo entero donde se almacenará el identificador de la subsección del menú administrador en la cual nos encontramos, leyendo el parámetro de tipo Querystring “idadm”.

Acto seguido se declara una cadena de caracteres “str”, a la que se irá añadiendo una serie de etiquetas. Cada uno de los elementos que forman el menú irá acompañado de las correspondientes etiquetas <li> y en la parte inicial y final del listado se sitúa la etiqueta <ul>, que servirá para declarar la lista de objetos. Dentro de las etiquetas <li> se ubican, las referencias a las páginas a las que se desea dirigir.

Cada una de ellas incluirá la etiqueta <a>, acompañada de la cadena que contiene la dirección de la página de destino. A continuación se incluye el nombre de la etiqueta que se mostrará, situada entre dos etiquetas , etiqueta para marcar bloques de texto.

Por otro lado, el método EscribeA comparará el identificador de cada subsección del menú con el valor almacenado en la variable idamin. Si dichos valores coinciden se añadirá en la etiqueta <a> de la cadena un identificador. Para que cuando apliquemos la hoja de estilos CSS dicho elemento de la lista se vea modificado respecto al resto y podamos destacarlo.

```
private string EscribeA(int valor, int idadm)
    {
        string str = "";
        if (valor == idadm)
            str += "id='current' ";
        return str;
    }
```

Se actúa de la misma forma con todos y cada uno de los elementos de la lista, y acto seguido, se cierra con otra etiqueta . El conjunto de toda esta cadena dará lugar al texto de un elemento de tipo literal, que conformará en sí mismo el menú admin.

6.2.2.3. Estilo CSS

El menú pues, estará formado por un conjunto de elementos con el contenido fijado por el código anterior, cuyo aspecto y posición quedará determinado por la hoja de estilos CSS correspondiente. Para ello, se situará en una determinada capa en la página maestra, a la que se aplicará dicho estilo. Añada este estilo a la Hoja de Estilo definida anteriormente (Fichero ~/App_Themes/Azul/Hoja_Estilo.css)

```
/*Estilo del Menú de Administración*/
#Menu_Admin /*Contenedor*/
{
width: 188px;
border-style: none none none none;
border-color: #BCD2E6;
border-width: 1px;
font-family: Tahoma;
text-align: center;
font-size: small;
}

#Menu_Admin li a
{
height: 32px;
voice-family: "\"}\"";
voice-family: inherit;
height: 24px;
text-decoration: none;
}

#Menu_Admin li a:link, #Menu_Admin li a:visited
{
color: Blue ;
font-weight:bold;
display: block;
background: url(Imagenes/Menu_Admin.JPG);
padding: 8px 0 0 0px;
}
#Menu_Admin li a:hover, #Menu_Admin li #current
{
color: White;
background: url(Imagenes/Menu_Admin.JPG) 0 -32px;
padding: 8px 0 0 0px;
}
#Menu_Admin ul
{
margin: 0;
padding: 0;
list-style: none;
}
```

6.2.3. Control Calendario Eventos

6.2.3.1. Descripción

El control calendario de eventos se mostrará en todas las páginas del sitio, al formar parte de la página maestra.

En él se van a mostrar todas aquellas fechas venideras en las que va a tener lugar algún acontecimiento (Evento). Dichas fechas aparecerán marcadas en negrita. Haciendo clic sobre ellas, se redirigirá a una pagina específica en la que se observarán todos los eventos que ocurrirán en la fecha en seleccionada.

También aparecerá remarcado, pero en un tono mas suave, la fecha del día actual. Además se podrá visualizar los eventos futuros de los próximos meses avanzando en el calendario.

Fig. 62 Control Calendario Eventos.

6.2.3.2. Desarrollo

Para implementar este control emplee un control Calendar propio de ASP.NET.

- Agregue un nuevo Control de usuario (denominado Calendario), en la carpeta controles
- En la vista Diseño, arrastre desde el Cuadro de herramientas un control de tipo **Calendar**
- Puede mejorar su apariencia el menú Tareas de Calendar, y elija un Formato.
- Establezca la propiedad SkinId al valor CalenMaster.

Para resaltar aquellas fechas en las que existirán eventos actuaremos de la siguiente forma. Estando en la vista Código (fichero Calendario.ascx.cs), declare una variable global del tipo string[] denominada arraydias que será la encargada de almacenar el conjunto de días para los que hay eventos programados. En el manejador del evento Load del control, es decir Page_Load, rellenaremos este array con el resultado de llamar al método GetFechasFuturas de la clase BLLEvento, por tanto recuerde siempre que se utilicen las clases BLL o BO, deberá hacer visibles los espacios de nombres adecuados.

```
using PortalWeb.BLL;
using PortalWeb.BO;

public partial class Controles_Calendario : System.Web.UI.UserControl
{
    protected string[] arraydias;

    protected void Page_Load(object sender, EventArgs e)
    {
        //Obtener las fechas en las que se llevarán a cabo los eventos
        arraydias = BLLEvento.GetFechasFuturas();

    }
```

En segundo lugar, defina manejador para el evento DayRender del control Calendar. Este evento se produce cuando se crea cada celda de fecha en el control Calendar, con lo que así podremos controlar el contenido y el formato de una cada una de ellas en el momento de crearla.

El manejador del evento comprobará mediante un bucle que la fecha de la celda que lo genera coincide con alguno de los valores del array de fechas obtenido anteriormente. De ser así, se personaliza la celda correspondiente, dándole formato negrita y subrayado, así como haciendo que su propiedad ToolTip sea igual a la fecha en cuestión. Por último se convierte dicha celda en un vínculo a una página donde se mostrarán los eventos programados para esa fecha.

```
//Se invoca cuando se crea cada una de las celdas
    protected void Calendar1_DayRender(object sender, DayRenderEventArgs e)
    {
        string path = Request.ApplicationPath; //Ruta de acceso a la raíz virtual de ASP .NET
        if (path == "/")
            path = "";

        for (int i = 0; i < arraydias.Length; i++)
        {
            string[] arrayFecha = e.Day.Date.ToString().Split(new char[] { ' ' });
```

```
            if (arrayFecha[0] == arraydias[i])
            {
                e.Cell.ToolTip = arraydias[i];
                e.Cell.Font.Bold = true;

                e.Cell.Text = "<a href='" + path + "/Controles/Eventos/Eventos.aspx?fecha=" +
arraydias[i] + "'>" + e.Day.DayNumberText + "</a>";
            }
        }
    }
```

En cuanto al aspecto del control, este podrá variar en función del tema seleccionado por cada usuario. Para ello se recurre al uso máscaras y se asignará un valor a la propiedad SkinId para poder personalizar el control. En este caso el Skin será “CalendarMaster”.

Hay que destacar que el control Calendar ofrece un amplio abanico de posibilidades a la hora de personalizar su aspecto. Se puede personalizar desde la cabecera del control, los fines de semana (pudiendo elegir cuando empiezan y acaban los mismos), el formato del día, etc.

6.2.4. Control Encuesta

6.2.4.1. Descripción

El control Encuesta al igual que los anteriores está contenido en la página maestra, de tal manera que se publicará en todas las páginas del sitio web.

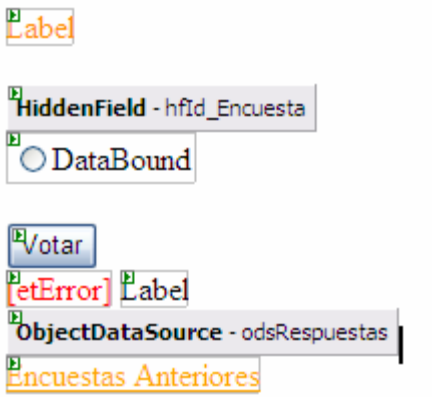
Su objetivo será exponer la encuesta activa más reciente. Para ello se mostrará el titulo y las posibles respuestas de la misma, y únicamente se podrá votar por una de ellas haciendo clic en el botón “Votar”, siempre y cuando nos el usuario se haya autenticado previamente.
Haciendo clic en la opción Encuestas Anteriores se redirigirá a una página donde aparecerán todas las encuestas existentes. Aquellas que sean antiguas estarán acompañadas por los resultados obtenidos, y por otro lado se tendrá la posibilidad de poder participar en las que aún no hayan sido cerradas.

6.2.4.2. Desarrollo

Para la creación del control se hará uso de diversos controles ASP.NET.

- Añada a la carpeta Controles una nueva carpeta llamada Encuestas
- Agregue a la carpeta Encuestas un control de usuario denominado Encuestas.

Estando en la vista diseño, coloque un control del tipo Label (etPregunta), para mostrar el titulo de la encuesta, adelantándose un poco a la aplicación de estilos, asigne a la propiedad CssClass el valor titulos y SkinID el valor de titulosmaster, con esto garantizamos que el texto cambie de apariencia al cambiar de tema . Además agregue un control HiddenField (hfId_Encuesta). Recuerde que este control representa un campo oculto que se utiliza para almacenar un valor no mostrado, que servirá para guardar el identificador de la encuesta.



Para mostrar la lista de respuestas asociadas a la misma se hará uso de un control RadioButtonList (rbList_Respuestas, CssClass = textos) así como de un objeto ObjectDataSource (odsRespuestas) asociado al RadioButtonList. Por otro lado, existe un control Button (btVotar, CssClass = textoscenter) para validar la opción escogida, un par de etiquetas (etError y etMensaje, Visible = False) para mostrar mensajes al usuario, asigne a la propiedad CssClass de ambas etiquetas el valor texos, finalmente agregue un control LinkButton (lnkEncuestasAnteriores, SkinId = titulosmaster, CssClass = textos) que permitirá visitar una página donde se podrá consultar el resto de encuestas.

Primeramente configure el origen de datos del objeto ObjectDataSource, seleccionando como objeto comercial PortalWeb.BLL.BLLRespuestaEncuesta. A continuación seleccione como método asociado con la operación SELECT, el método ListByIdEncuesta. Establezcas como origen del parámetro (id) el control huid_Encuesta, de tal manera que dependiendo del identificador almacenado en el campo oculto se podrá visualizar las respuestas asociadas a una u otra encuesta.

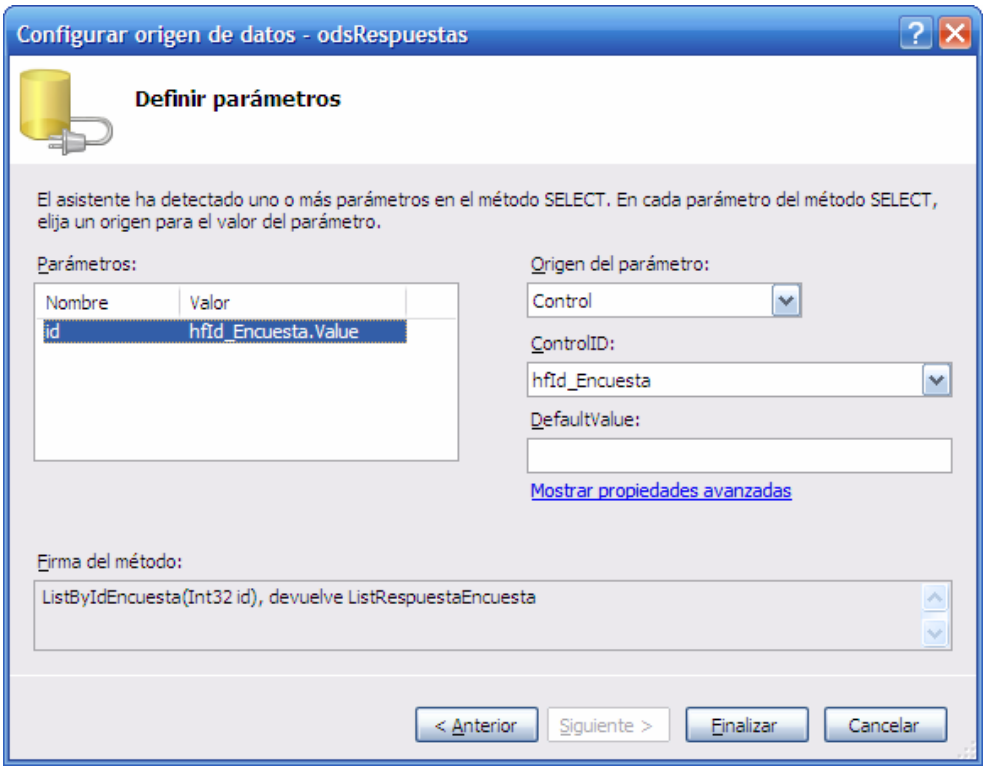
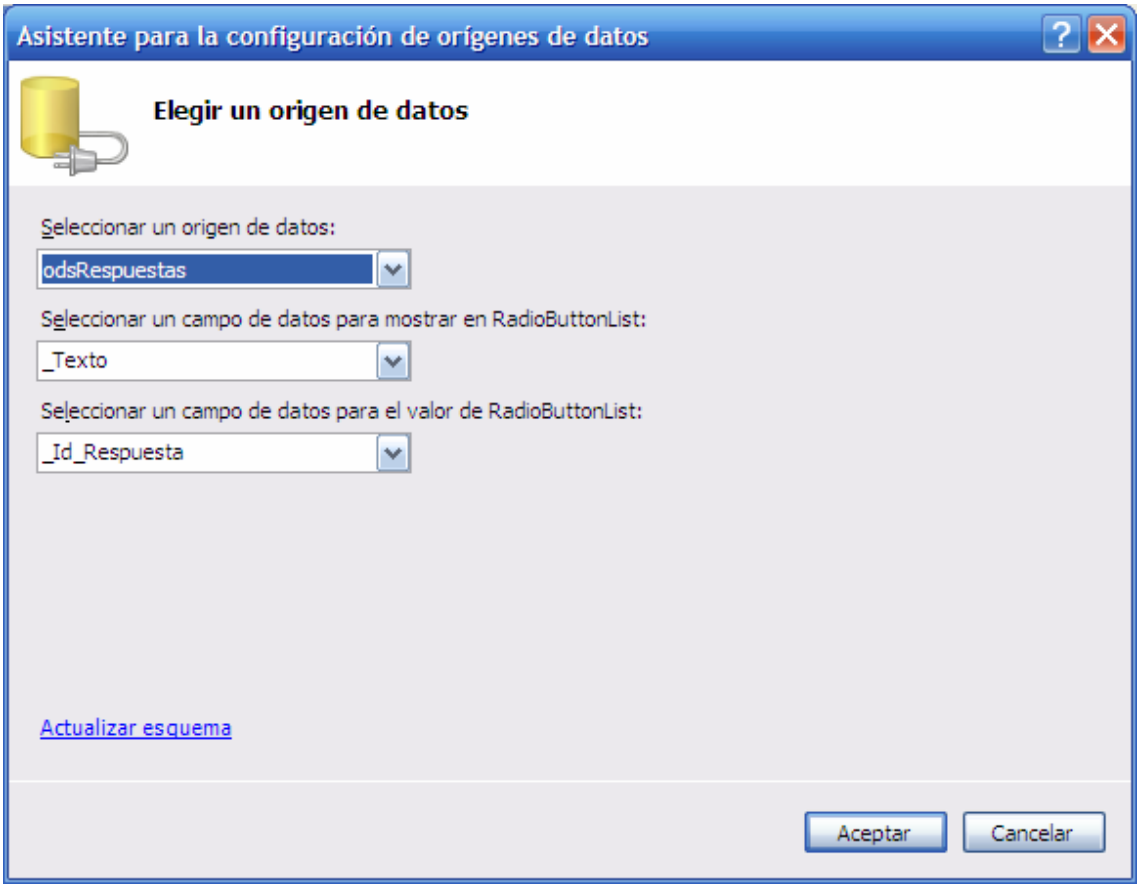


Fig. 63 Configurando origen de datos del control Encuesta.

Ahora configure las tareas del control rbList_Respuestas, de tal manera que su origen de datos sea el objeto odsRespuestas, configurado anteriormente, se mostrará el campo Texto, por tanto seleccione la propiedad _Texto y el campo _Id_Respuesta, será el valor del objeto rbList_Respuestas.

Además cree manejador para el evento Clic del Button (btVotar) que se denominará btVotar_Click. En primer lugar, el manejador crea una variable del tipo RespuestaEncuestaUsr que almacenará los datos necesarios para registrar el voto emitido. Almacena en el campo Id_Encuesta de la misma, el identificador de la encuesta actual y guarda en una variable de tipo entero el valor del RadioButton seleccionado.



Seguidamente se comprueba si el usuario está o no autenticado; si no lo está, la etiqueta etError pasaría a ser visible indicando que el usuario ha de estar registrado para poder realizar la operación. En el caso de que el usuario estuviera autenticado, hay que comprobar si ha votado con anterioridad sobre la misma encuesta. Para ello se crear un variable de tipo booleado que almacena el resultado de llamar al método UsrHaVotado_IdEncuesta de la clase BLLRespuestaEncuestaUsr. Si esta variable es igual a true, la etiqueta etEerror pasaría a ser visible, en este caso indicando que únicamente se puede votar una vez. Si por el contrario la variable es igual a false, es decir, el usuario no ha votado todavía, se almacenaría en el campo Usuario del objeto de tipo RespuestaEncuestaUsr el nombre del usuario actual, y en el campo Id_Respuesta el identificador de la respuesta seleccionada (Recuerde que para acceder a los campos se debe hacer uso de las propiedades definidas en los objetos de negocio. A continuación se realiza una llamada al método AddUpdate de la clase BLLRespuestaEncuestaUsr, para guardar todos los datos, y la etiqueta etMensaje pasaría a ser visible indicando que la operación ha sido completada. Para poder utilizar la capa BLL y BO, haga visible los espacios de nombres correspondientes.

```
using PortalWeb.BLL;
using PortalWeb.BO;
……
……
protected void btVotar_Click(object sender, EventArgs e)
   {
      RespuestaEncuestaUsr mirespuestaUsr = new RespuestaEncuestaUsr();
      //Id de la encuesta en la que está votando
      mirespuestaUsr._Id_Encuesta = int.Parse(hfId_Encuesta.Value);
      //Opción de las respuestas seleccionada
      int idrespuesta = int.Parse(rbList_Respuestas.SelectedValue);
      //Indagar si el usuario está registrado en el portal
      string usuario = HttpContext.Current.User.Identity.Name;
      if (usuario == String.Empty)
      {
         etError.Visible = true;
         etError.Text = "Debes estar registrado";
         rbList_Respuestas.SelectedIndex = -1;
      }
      else
      {
         //Llamar al método UsrHaVotado_IdEncuesta, para indagar si el usuario
         //ya ha votado sobre esa encuesta
         bool havotado =
BLLRespuestaEncuestaUsr.UsrHaVotado_IdEncuesta(mirespuestaUsr._Id_Encuesta, usuario);
         if (havotado)
         {
            etError.Visible = true;
            etError.Text = "Solo puedes votar una vez";
            rbList_Respuestas.SelectedIndex = -1;
            etMensaje.Visible = false;
         }
         else
         {
            mirespuestaUsr._Usuario = usuario;
            mirespuestaUsr._Id_Respuesta = idrespuesta;
            BLLRespuestaEncuestaUsr.AddUpdate(mirespuestaUsr);
            rbList_Respuestas.SelectedIndex = -1;
            etMensaje.Visible = true;
            etMensaje.Text = "Tu voto ha sido registrado";
         }
      }
   }
```

Por último, para que el control adquiera la funcionalidad buscada se debe primeramente configurar el manejador del evento Load del mismo (Page_Load). Este manejador almacena en una variable de tipo Encuesta el resultado de la llamada al método SelectGetUltima, es decir, la encuesta activa más reciente. Si dicho método devuelve null, significa que no hay ninguna encuesta activa con lo que el botón Votar se ocultaría y en la etiqueta correspondiente al titulo se leería el mensaje "*No hay encuestas activas*".

Si el método retorna una encuesta, almacena en el campo oculto el identificador de la misma y establece la propiedad Text de la etiqueta etPregunta igual al valor del campo Pregunta.

```
protected void Page_Load(object sender, EventArgs e)
    {
        Encuesta miencuesta = new Encuesta();
        //Obtener la última encuesta
        miencuesta = BLLEncuesta.SelectGetUltima();

        //Si no hay encuestas recientes
        if (miencuesta != null)
        {
            //Obtener los datos de la encuesta
            hfId_Encuesta.Value = miencuesta._Id_Encuesta.ToString();
            etPregunta.Text = miencuesta._Pregunta;
            odsRespuestas.SelectParameters["id"].DefaultValue =
miencuesta._Id_Encuesta.ToString();
        }
        else //De lo contrario
        {
            btVotar.Visible = false; //Ocultar el botón Votar
            etPregunta.Text = "No hay Encuestas Activas";
        }
    }
```

6.2.4.3. Estilo CSS

A lo largo del desarrollo del control Encuesta, se ha establecido la propiedad CssClass de varios objetos, esta propiedad define el nombre de la clase CSS aplicada al control, estas clases son definidas dentro de un archivo tipo Hoja de estilo, por tanto añada al fichero Hoja_Estilo.css definido anteriormente la definición de las clases aquí ocupadas.

```
.textoscenter
{
        font-family: Tahoma;
        text-align:center;
        font-size: small;
}

.titulos
{
        font-size: medium;
        text-transform: uppercase;
        color: black;
        font-family: Tahoma;
        font-variant: small-caps;
        font-weight: bold;
```

```
        text-align: left;
}

.textos
{
        color: black;
        font-family: Tahoma;
        text-align: justify;
        font-size: small;
}
```

6.2.5. Control Noticias más comentadas

6.2.5.1. Descripción

El control Noticias más comentadas se visualizará en todas las páginas del sitio, ya que forma parte del conjunto de controles que se encuentran en la página maestra.

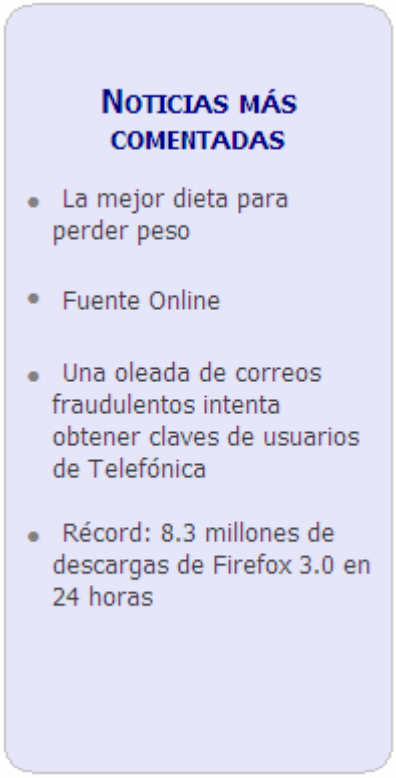


Fig. 64 Control Noticias más comentadas.

Su finalidad es mostrar aquellas cinco noticias que reúnan el mayor número de opiniones. Se trata sin duda, de un control dinámico, que irá variando conforme los usuarios vayan insertando opiniones. Haciendo clic sobre cualquiera de ellas, se acudirá a la noticia en sí, para poder leer toda la información relacionada.

6.2.5.2. Desarrollo

Para la creación de este control únicamente se empleará un objeto ObjectDataSource (ods1) y un control enlazado a datos DataList (DataList1) cuyo origen de datos será el propio ObjectDataSource (ods1).

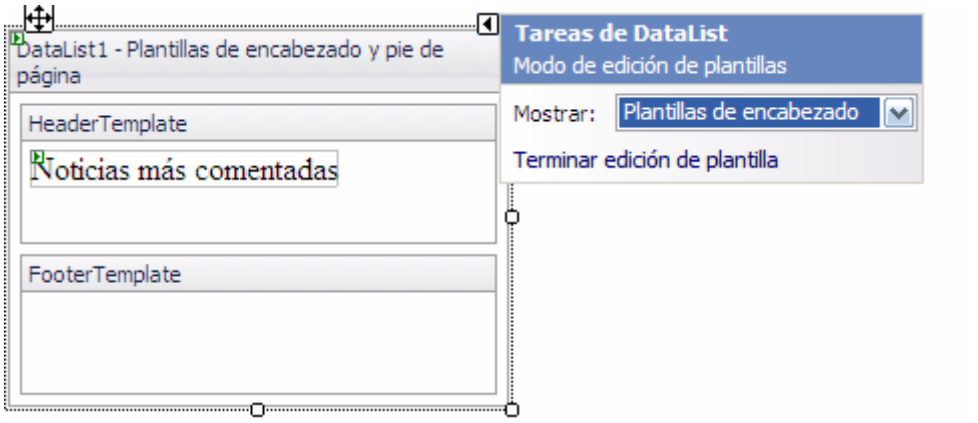
Realice los siguientes pasos:

- En la carpeta Controles, añada una nueva carpeta llamada Noticias.
- Dentro de la carpeta Noticias, añada un nuevo control de usuario denominado NoticiasMasComentadas.

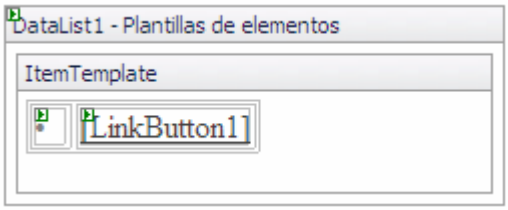
Estando en la vista diseño, agregue desde el Cuadro de herramientas, un objeto ObjectDataSource y otro DataList. Primero configure el origen de datos del objeto ObjectDataSource, seleccionando como objeto comercial la clase BLLNoticia y como método asociado a la operación SELECT ObjectDataSource el método ListMasComentadas.

A continuación establezca como origen de datos del control DataList el ObjectDataSource (ods1) y proceda a editar sus plantillas. Interesará la plantilla ItemTemplate y la plantilla HeaderTemplate.

En la plantilla de encabezado coloque una etiqueta cuya propiedad Text corresponderá con el titulo del control ("**Noticias más comentadas**"), y establezca la propiedad CssClass a titulos y SkinId a titulosmaster.



En la plantilla ItemTemplate cree una tabla con una única fila y dos columnas. En la primera columna coloque una etiqueta cuya propiedad Text se corresponderá con un carácter para representar guiones y CssClass= etiquetas. En la segunda columna sitúe un control LinkButton (CssClass = etiquetaspequeñas) cuyas propiedades Text y PostBackUrl se enlazarán a datos. La primera la enlazaremos al campo _Título, mientras que la segunda se enlaza al campo _Id_Noticia. Además se debe configurar el formato de ésta última para que el identificador de la noticia sea pasado como QueryString a la página de destino. De esta manera cuando se pulse sobre uno de los LinkButtons se redirigirá a una página con toda la información asociada a la noticia seleccionada.



Para hacer esto siga los siguientes pasos:

- Abra el menú *Tareas de LinkButton*
- Clic en *Editar DataBinding*
- Seleccione la propiedad enlazable *Text*
- Marque el botón de radio *Enlace a campo:*
- Seleccione de la lista desplegable *Con enlace a:* la propiedad *_Título*
- Clic en Aceptar

La propiedad PostBacUrl se puede establecer desde diseño, pero aquí se configura desde código, pues aún no se cuenta con la estructura completa del sitio Web, a continuación se muestra este código y el generado por el diseñador al configurar la propiedad Text del objeto LinkButton.

```
<asp:LinkButton ID="LinkButton1" runat="server" CssClass="etiquetaspequeñas"
        PostBackUrl='<%# Eval("_Id_Noticia", "~/Controles/Noticias/Noticia.aspx?id={0}") %>'
        Text='<%# Eval("_Título") %>' ForeColor="#404040">
</asp:LinkButton>
```

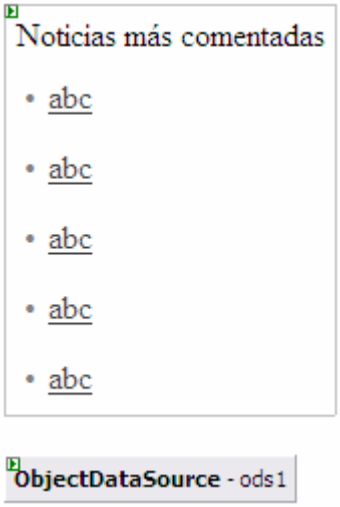


Fig. 65 Diseño Control Noticias más comentadas.

6.2.5.3. Estilo CSS

Siguiendo con la edición de las Hojas de Estilo en Cascada, añada al fichero Hoja_Estilo, las clases aquí utilizadas.

```
.etiquetas
{
        color: black;
        font-family: Tahoma; /*text-align: justify;*/
        font-size: medium;
}

.etiquetaspequeñas
{
        color: black;
        font-family: Tahoma; /*text-align: justify;*/
        font-size: small;
        padding-left: 5px;
        text-align: left;
}
```

6.2.6. Control Buscador

6.2.6.1. Descripción

Para favorecer la búsqueda de contenido en el sitio Web por parte de los usuarios, se creará el control Buscador. Este control se va a encargar de llamar a un servicio web que nos proporciona Google. El usuario deberá introducir el texto que desee buscar y a continuación presionar el botón Buscar. Esto llevará al usuario a una página donde se mostrará el resultado de la búsqueda, y desde ahí se podrá dirigir a la página que ofrece la noticia.

6.2.6.2. Desarrollo

Para la creación de este control añada una nueva subcarpeta (Google) dentro de la carpeta Controles, en la cual se deberá incluir un control de usuario (Google.ascx).

Añada un etiqueta con su propiedad Text = “Buscador”, a continuación añada una caja de texto, destinada a recoger la información que desea buscar el usuario, inserte a continuación una imagen alusiva al buscador Google y finalmente añada un botón, cuya propiedad Text= “Buscar”, defina un grupo de validación para la caja de texto y para el control.

Fig. 66 Diseño del control Buscador.

Defina el controlador del evento clic del botón Buscar de la siguiente manera:

```
protected void btBuscar_Click(object sender, EventArgs e)
    {
        Response.Redirect("~/Controles/Google/MostrarResultados.aspx?ambito=" +
"&cadena=" + TextBox1.Text);
    }
```

Este controlador, nos envía a una nueva página (que posteriormente será explicada), pasándole como cadena el texto que el usuario ha introducido en la caja de texto.

6.3. Página Maestra

6.3.1. Descripción

La creación de una página maestra tiene como objetivo dar un aspecto uniforme a todas las páginas del sitio Web. En esta aplicación, la estructuración estará basada en un diseño clásico de tres columnas con una cabecera.

Como se observa en la **Fig. 67**, las columnas laterales estarán destinadas a albergar los controles comunes a todas las páginas anteriormente descritos, y la columna central, proporcionará los contenidos específicos para cada página mediante el control *ContentPlaceHolder*.

La cabecera, que también será común a todas las páginas, contendrá el logo del sitio Web, la fecha actual y el menú principal de navegación.

En la columna situada a la izquierda se ubicará el control destinado al acceso de usuarios, el control que mostrará las noticias más comentadas y el menú de administración del sitio, que únicamente será visible en caso de que el usuario que inicie sesión tenga el rol de “Administrador”.

Por otro lado, en la columna situada a la derecha se colocará el control buscador dedicado a realizar búsquedas en Google desde el sitio Web, el control destinado a mostrar la encuesta activa más reciente y el control Calendario de eventos.

Cada uno de los controles de las columnas laterales se ubicará a su vez dentro de un panel, el cual presentará un aspecto de esquinas redondeadas.

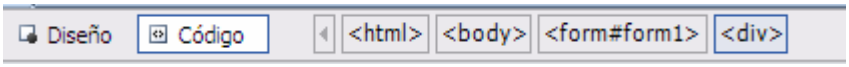
Fig. 67 Estructura de las páginas.

6.3.2. Desarrollo

En primer lugar agregue una página maestra en el directorio del sito Web:

- Clic derecho sobre el directorio del sitio (*http://localhost/PortalWeb*)
- Seleccione *Agregar un nuevo elemento …*
- Seleccione la plantilla *Página Principal*
- Asígnele como nombre *Página_Maestra*
- Clic en Aceptar

Para mayor facilidad en el diseño de la página elimine por el momento el control ContentPlaceHolder, a continuación y estando en modo Código, cree las siguientes capas:




```
<form id="form1" runat="server">
      <div id="container">  <!-- Contenodor principal -->
         <div id="header">  <!-- Cabecera -->
         </div> <!-- Cabecera -->
         <div id="outer"> <!-- El esto del cuerpo-->
            <div id="inner"> <!--Alberga la estructura principal de la página -->
               <div id="left">
               </div> <!--Panel de la izquierda-->
               <div id="right">
               </div> <!--Panel derecho -->
               <div id="central">
               </div> <!--Panel central -->
            </div>
         </div>
      </div>
   </form>
```

Existe una capa principal denominada “container” que se subdividirá en otras dos capas, una específica para la cabecera, la capa “header”, y la otra para el resto, “outer”. Esta última capa a su vez contendrá otra capa, denominada “inner”, y será dentro de ésta donde incluyamos las tres capas correspondientes a las tres columnas en las que se divide la estructura principal de la página. El orden de estas tres últimas capas será el siguiente: la izquierda, (“left”), la derecha (“right”) y la central (“central”).

Es hora de definir los estilos que se aplicarán a las capas creadas anteriormente, por tanto edite el archivo Hoja_Estilo de la siguiente manera:

```
#container
{
width: 950px;
padding: 0px;
margin: 0px;
margin-left: auto;
margin-right: auto;
}
#header
{
position: relative;
height: 100px;
width: 100%;
background-color: #ffffff;
margin-bottom: 22px;
_margin-bottom: 7px;
top: 0px;
left: 0px;
}
#outer
{
border-left: solid 203px #ffffff;
border-right: solid 205px #ffffff;
background-color: #ffffff;
}
#inner
{
margin : 0px;
width: 100%;
}
#left
{
width: 200px;
float: left;
position: relative;
margin-left: -203px;
_margin-left: -203px;
margin-right: 1px;
text-align:center;
}
#right
{
width: 200px;
float: right;
position: relative;
margin-right: -205px;
_margin-right: -205px;
margin-left: 1px;
text-align:center;
}
#central
{
position: relative;
margin: 0px;
width: 540px;
}
```

Ahora se irán definiendo cada una de las secciones que componen la página maestra:

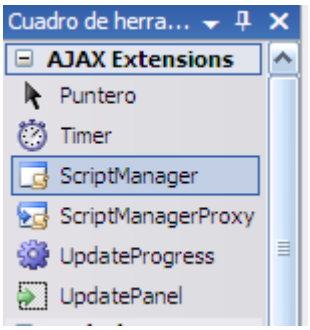
Cabecera

La división cabecera ( ), contendrá una tabla con dos tres columnas y dos filas, que serán combinadas de la siguiente forma

El código correspondiente es el siguiente:

```
<div id="header">  <!-- Cabecera -->
            <table width="100%">
                <tr>  <!-- Logo -->
                    <td rowspan="2" style="width: 200px" valign="top">
                    <td>
                    </td>
                    <td style="height: 43px; width: 147px;">
                    </td>
                </tr>
                <tr>
                    <td style="height: 20px" colspan="2" rowspan="2">
                    </td>
                </tr>
             </table>
</div> <!-- Cabecera -->
```

En la celda de la izquierda, arrastre desde el Cuadro de herramientas, un control de tipo Image, y asigne a la propiedad SkinId de este el nombre “logo”, así se define la imagen que se visualizará cuando de aplique uno u otro archivo de máscara.



En la celda superior central coloque un control de ScriptManager para poder realizar todas las operaciones con los controles de ASP.NET AJAX en las diferentes páginas afectadas por la masterpage.

Fig. 68 Control AJAX ScriptManager.

En la celda superior derecha incluya una capa denominada “masterfecha”, utilizada para dar un estilo determinado al contenido de la propia celda, y en su interior coloque un control del tipo Literal (litFecha). El código es el siguiente:

```
<td style="height: 43px; width: 147px;">
        <div id="MasterFecha">
                <asp:Literal ID="litFecha" runat="server"></asp:Literal>
         </div>
</td>
```

La propiedad Text de este literal se establece en el manejador del evento Load de la página, el método Page_Load, y corresponderá con la fecha actual del sistema en formato texto.

```
protected void Page_Load(object sender, EventArgs e)
   {
      litFecha.Text = DateTime.Now.ToLongDateString().ToString();
   }
```

A la celda “masterfecha” se le aplica el siguiente estilo, el cuál se encontrará en la hoja de estilos CSS:

```
#MasterFecha
{
text-align: right;
font-size: 0.9em;
margin: 0 5px 0 0;
font-family: Tahoma;
}
```

Por último en la celda inferior de la tabla incluya otra capa denominada “MenuPrincipal” que dará formato al contenido de la misma. Esta celda albergará al control Menú_Principal, por lo que el estilo aplicado al menú se verá descrito según el estilo definido en la hoja CSS para la capa “MenuPrincipal”, el cual fue creado en la sección correspondiente al control Menú Principal.

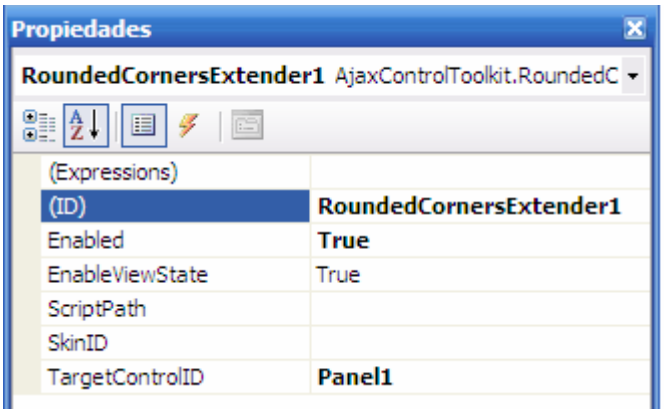
Para incluir el control de usuario la capa MenuPrincipal, estando en el Explorador de soluciones, simplemente selecciónelo y arrástrelo al lugar deseado.

Columnas laterales

En la columna izquierda coloque tres paneles en los que posteriormente situará el control de inicio de sesión, el Menú de Administración, y el control de Noticias más Comentadas creados anteriormente.
Los tres paneles tendrán una anchura de 192px, su propiedad “Height” déjela sin especificar, la propiedad HorizontalAling establézcala a “Center”, y, por otro lado, su propiedad SkinId será igual a “Panel” para en el archivo de máscaras especificar su color de fondo.

En la columna derecha coloque tres paneles con las mismas características que los anteriores. En ellos situará los controles Buscador, Encuesta y Calendario Eventos anteriormente creados. Los paneles los puede añadir y configurar desde vista diseño o desde código.

A todos los paneles se les va a aplicar un estilo de esquinas redondeadas mediante el extender RoundedCornerExtender de Ajax, por lo que finalmente añada un extender por cada panel y los configúrelo de la siguiente manera.



La propiedad TargetControlID de cada extender corresponderá con cada uno de los paneles. Para personalizar dichas esquinas acudiremos a las propiedades del extensor visualizadas en el menú de propiedades de cada panel, ver **Fig. 69**.

De esta forma puede elegir el color del borde así como el color de las esquinas, el radio de las mismas y qué esquinas se verán afectadas por este efecto.

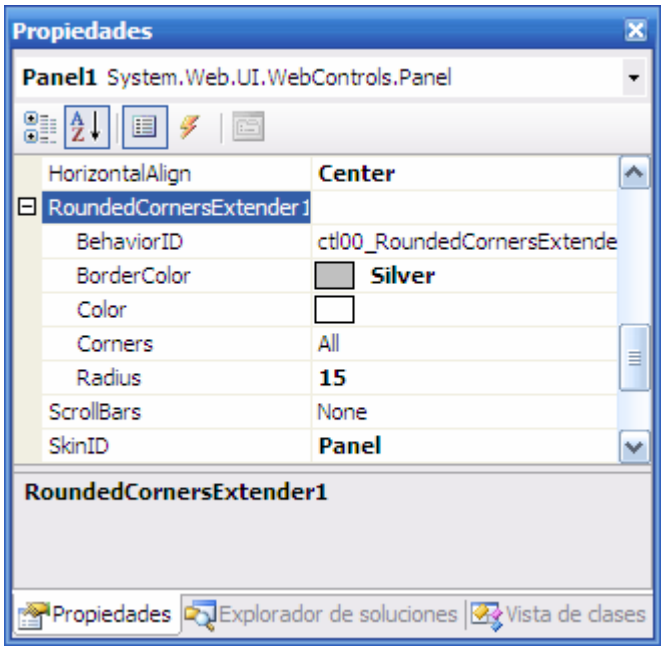


Fig. 69 Propiedad RoundedCornersExtender.

El control Menú Administradores, sólo será visible para usuarios que tengan el rol Administradores, que será explicado más adelante, por el momento, este control no deberá ser visible, por tanto añada las siguientes líneas al control del evento Load de la página maestra.

```
…..
…..
Panel2.Visible = HttpContext.Current.User.IsInRole("Administradores");
Panel1.Focus();
```

Para poder ver el efecto de las hojas de estilo de las propiedades SkinID, de los controles creados, añada a la carpeta Azul un Nuevo elemento de tipo Archivo de máscara, utilizado para definir un tema en ASP .NET y edítelo de la siguiente manera:

```
<asp:Hyperlink runat="server" SkinId = "titulosmaster" ForeColor = "Navy"/>
<asp:Linkbutton runat="server" SkinId = "titulosmaster" ForeColor = "Navy"/>
<asp:Image runat="server" SkinId = "Imagenes" BorderColor = "Lavender"/>
<asp:Image runat="server" SkinId = "Logo" ImageUrl="Imagenes/Logo.png"/>
<asp:Panel runat="server" SkinId = "Panel" BackColor = "Lavender"/>
<asp:Label runat="server" SkinId = "titulosmaster" ForeColor = "Navy"/>

<asp:Calendar runat="server" SkinId = "CalenMaster" BackColor="White"
BorderColor="#3366CC"  ForeColor="#003399">
<SelectedDayStyle BackColor="#009999" Font-Bold="True" ForeColor="#663399" />
    <SelectedDayStyle BackColor="#009999" Font-Bold="True" ForeColor="#CCFF99" />
        <TodayDayStyle BackColor="#99CCCC" ForeColor="White" />
```

```
        <SelectorStyle BackColor="#99CCCC" ForeColor="#336666" />
        <WeekendDayStyle BackColor="#CCCCFF" />
        <OtherMonthDayStyle ForeColor="#999999" />
        <NextPrevStyle  ForeColor="#CCCCFF" />
        <DayHeaderStyle BackColor="#99CCCC" ForeColor="#336666" Height="1px" />
        <TitleStyle BackColor="#003399" BorderColor="#3366CC" Font-Bold="True"
ForeColor="#CCCCFF"  />
</asp:Calendar>
```

Este archivo al igual que el archivo de Hoja de Estilo se explicarán con posterioridad y serán ampliados a lo largo del desarrollo del Portal, ahora para que estos estilos tengan efectos el Portal deberá aplicar dicho Tema (Azul), momentáneamente defina en el archivo web.config dicho tema y asígnele como valor Azul, el creado hasta el momento, más adelante verá que no es necesario aplicar el tema explícitamente sino que se hará dinámicamente según el elegido por el usuario. (*Ver Capítulo1, Sección Temas y Máscaras de ASP .NET 2.0)*

```
<pages theme ="Azul">

</pages>
```

Añada a la capa center, un control de tipo ContentPlaceHolder, hasta aquí termina la edición de la página maestra, puede ejecutar la aplicación y observará ya algunos resultados, siempre y cuando tenga datos en la base de datos.

6.4. Página Default

6.4.1. Descripción

Esta página dará la bienvenida a los usuarios que accedan al portal contendrá en su parte central las noticias más recientes. Se mostrarán tan sólo un número reducido de ellas, y bastará con hacer clic en el título de cualquiera de las mismas para remitirse a la noticia en sí. También se puede observar al pie de cada noticia el número de opiniones vertidas por los usuarios. Pinchando en ese enlace, se dirigirá directamente hasta dichos comentarios, que se sitúan en la parte inferior de la noticia. Se observa también un enlace que nos redirigirá a una página donde se pondrán buscar las noticias anteriores.

Se dispone dispondremos de la posibilidad de que los navegantes se suscriban al canal de noticias RSS y poder así estar al corriente de todas las novedades que se produzcan, desde su propio lector de feeds RSS. Para ello, deberán hacer clic en el icono RSS situado en la parte superior de la página.

Fig. 70 Página Default.aspx

6.4.2. Desarrollo

Esta página está formada principalmente por un control de usuario llamado NoticiasPortada.ascx. En primer lugar se describe la creación de dicho control para acto seguido pasar a detallar el desarrollo de la página en sí.

6.4.2.1. Control NoticiasPortada

Como se ha mencionado mediante este control se presentan las noticias más recientes. Para mostrar estos datos utilizaremos un control de enlace a datos del tipo DataList que estará enlazado a un ObjectDataSource.

Agregue un control de usuario dentro de la carpeta ~/Controles/Noticias denominado NoticiasPortada.ascx, arrastre desde el cuadro de herramientas un control ObjectDaraSource (ObjectDataSource1) y un control DataList (DataList1).

Primero configure el origen de datos del objeto ObjectDataSource y eligiendo como objeto comercial BLLNoticia y como método del objeto comercial asociado a la operación SELECT el método SelectListPortada.

A continuación, seleccione origen de datos del control DataList1, el control ObjectDataSource1 y fije el resto de propiedades según la **Fig. 71**.

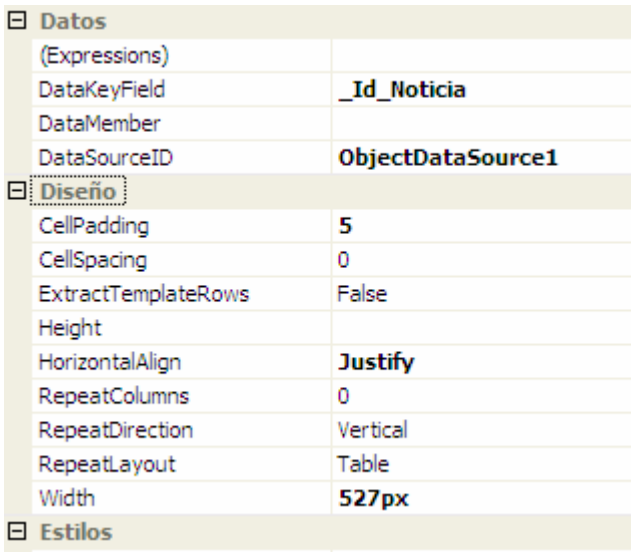
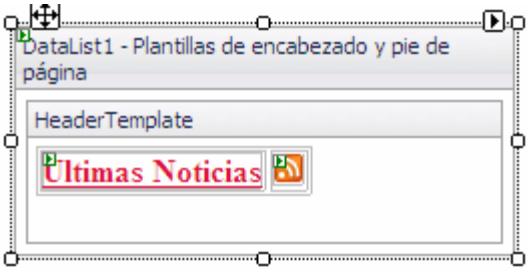


Fig. 71 Seleccionar un origen de datos para un control DataList.

El siguiente paso es editar las plantillas del DataList. Se editará plantilla de cabecera y la plantilla ItemTemplate. En la primera, añada una tabla formada por una fila y dos columnas, en la columna de la izquierda situaremos una Label que dará titulo a la página con el texto “Últimas noticias” (CssClass = etiquetas, SkinID = titulosmaster), y en la otra columna coloque un control

Hyperlink. Este hipervínculo permitirá a los usuarios suscribirse a un canal RSS que contendrá la información referente a las últimas noticias publicadas. Su propiedad ImageUrl estará ligada al path del icono de RSS y su propiedad NavigateUrl será igual a la página ~/Controles/Noticias/rss20.aspx, en esta página se podrá acceder a los datos de este control en formato rss. Esto lo puede hacer desde la vista de diseño, el código correspondiente es el siguiente:

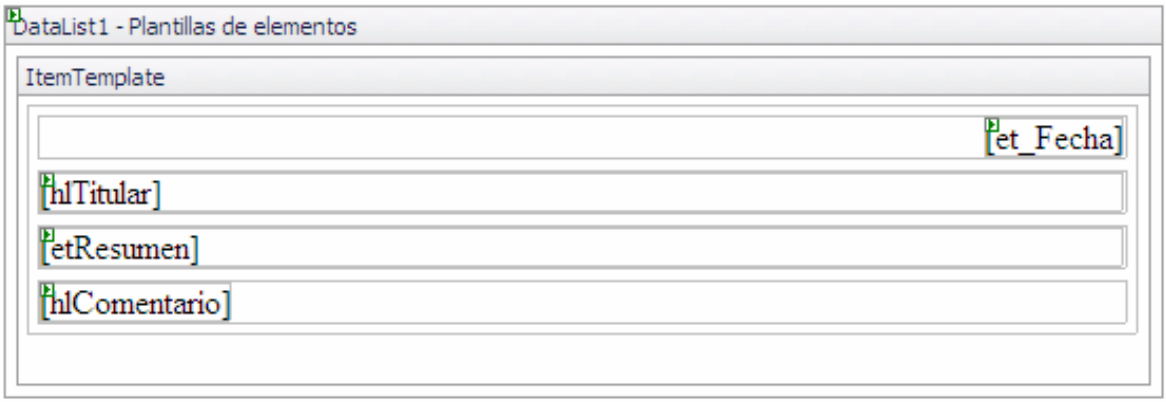



```
<HeaderTemplate>
      <table>
         <tr>
            <td>
               <asp:Label ID="Label3"
runat="server" CssClass="etiquetas" Font-
Bold="True" Font-Underline="True"
                  ForeColor="Crimson"
Text="Últimas Noticias"
SkinID="titulosmaster"></asp:Label></td>
            <td align="center" valign="middle">
               <asp:HyperLink ID="HyperLink3"
runat="server" ImageUrl="~/Imagenes/feed-
icon.png"

NavigateUrl="~/Controles/Noticias/RSS20.aspx"
ToolTip="Descarga nuestras noticias en
formato RSS"></asp:HyperLink></td>
         </tr>
      </table>
</HeaderTemplate>
```

Observe la propiedad *ImageUrl* del control *HyperLink*, se establece a una imagen **feed-icon.png** dentro de una carpeta *Imagenes* colgando del directorio raíz del sitio, por tanto cree este carpeta, descargue la imagen desde el sitio oficial de la asignatura guárdela en esta carpeta y de clic derecho sobre la carpeta *Imagenes*, seleccione la opción *Agregar elemento existente…*, seleccione la imagen descargada recientemente y de clic en *Agregar*, esto lo deberá hacer de aquí en adelante para todas las imágenes que necesitará en el desarrollo del Portal.

La propiedad *NavigateUrl* de este mismo control, se enlaza con una página llamada **RSS20.aspx**, que posteriormente será explicada.

Seguidamente edite la plantilla ItemTemplate. En ésta, agregue una tabla para disponer de manera más estructurada los diferentes elementos. La tabla se compondrá de una columna y cuatro filas, su anchura será de 523 px y el espaciado entre celdas será de 5px.



La anchura de la primera celda se establece al 100% (Style = WIDTH:100%), mientras que en el resto de las celdas dejaremos esta propiedad sin establecer. Además en la primera celda fijaremos el alineamiento a la derecha (Align = right) y en la última a la izquierda (left).

A continuación agregue los siguientes elementos: dos controles Label (et_Fecha y etResumen) uno en la primera y otro en la tercera celda, y dos controles HyperLink, uno en la segunda y otro en la cuarta celda (hlTitular y hlComentario). La etiqueta et_Fecha tendrá un ancho del 100% mientras que en etResumen y hlTitular el ancho será de 514 píxeles. Por otro lado a cada elemento le aplíquele una clase CSS para personalizar su aspecto en cuanto al tipo de fuente, el tamaño y el color.

Control	CssClass	SkinID
et_Fecha	textos	titulosmaster
hlTitular	titulos	titulosmaster
etResumen	textos	-----
hlComentario	resumenes	titulosmaster

Por tanto añada al fichero Hoja_Estilo lo siguiente:

```
.resumenes
{
        color: black;
        font-family: Tahoma;
        text-align: justify;
        font-size: small;
        font-style: italic;
        font-variant: normal;
}
```

Configure cada uno de estos elementos de la siguiente forma: Enlace la propiedad Text de la etiqueta et_Fecha al campo de datos _Fecha (Expresión de código: Eval("_Fecha", "{0:d}")), la propiedad Text del hipervínculo hlTitular al campo de datos _Título y su propiedad NavigateUrl con el campo de datos _Id_Noticia. Además establezca como formato de la misma el siguiente código para que al hacer clic sobre él el usuario sea redirigido a una página con más información sobre la noticia, ver en la **Fig. 72**

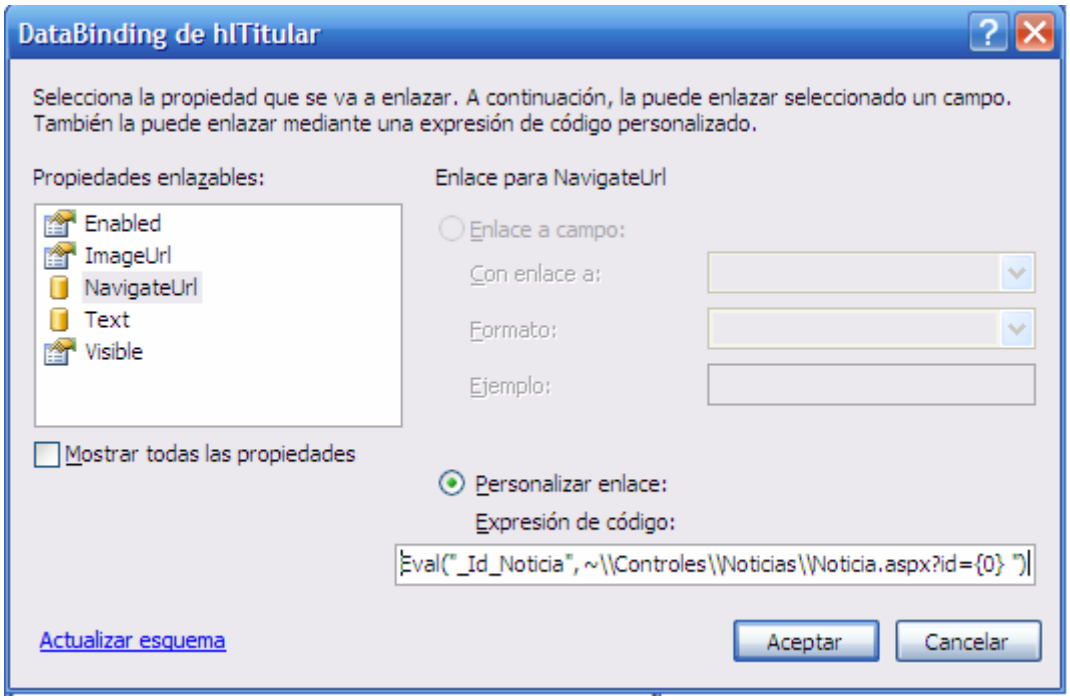


Fig. 72 Control hlTitular.

La propiedad Text de la etiqueta etResumen se enlazará con el campo de datos _Resumen, y por último en el hipervínculo hlComentario se enlazan las propiedades Text y NavigateUrl con el campo _Id_Noticia de la siguiente manera:

```
<td align="left">
      <asp:HyperLink ID="hlComentario" runat="server" CssClass="resumenes" Text='<%#
GetNOpiniones(Eval("_Id_Noticia")) %>'
      NavigateUrl='<%#
Eval("_Id_Noticia","~\\Controles\\Noticias\\Noticia.aspx?&id={0}&#Opiniones") %>'
SkinID="titulosmaster">
       </asp:HyperLink>
</td>
```

Como se observa, la propiedad Text va ligada al método GetNOpiniones, este método deberá definir en el fichero NoticiasPortada.ascx.cs de la siguiente manera:

```
protected string GetNOpiniones(object idnoticia)
   {
      int opiniones = BLLOpinión.NumOpinionesByIdNoticia((int)idnoticia);
      if (opiniones == 0)
         return ("Ninguna opinión");
      else if (opiniones > 1)
         return (opiniones.ToString() + " opiniones");
      else
         return (opiniones.ToString() + " opinión");
   }
```

Este método devuelve una cadena de caracteres indicando el número de opiniones que existen sobre una noticia. Dicha cadena está formada por el entero devuelto por el método NumOpinionesByIdNoticia, es decir el número de opiniones, más la palabra “opinión” en el caso de haber únicamente una opinión o la palabra “opiniones” si hay más de una. En el caso de que todavía no haya ninguna opinión el método devuelve la cadena de caracteres “Ninguna opinión”. Esos serán por tanto, los diferentes mensajes que muestre el hipervínculo hlComentario en cada uno de los casos.

Por otro lado, se ha añadido la cadena de texto “&#Opiniones” a la expresión de la dirección Url a explorar para indicar que dentro de dicha página se desea ir a la sección denominada Opiniones. Para ello, más tarde en la página destino (Noticia.ascx) se incluirá el siguiente Tag antes del apartado de opiniones.

```
<a id="Opiniones"></a>
```

6.4.2.2. Página Default.aspx

En primer lugar elimine la página Default.aspx, que se crea automáticamente junto con el sitio, pues se desea que esta página tome como página principal, la página maestra hasta ahora creada.

Añada un nuevo formulario Web (Default.aspx) y elija la opción "Seleccionar la página principal", para en la ventana siguiente escoger la página que se ha creado "Página_Maestra.master". Cada vez que cree una página .aspx repita este procedimiento para hacer que todas las páginas del sitio tengan un aspecto uniforme.

Una vez hecho esto añada un Panel en la región de contenido de 536 píxeles de ancho, para ajustarse de esta manera a la columna central, y su propiedad HorizontalAling a "Center". Seguidamente añada un control RoundedCornersExtender que aportará un diseño de esquinas redondeadas al panel, por lo tanto la propiedad TargetControlId del mismo será el panel (Panel1), igual a como se explicó en diseño de la Página Maestra, estos pasos también se deberán repetir para la mayoría de páginas .aspx de aquí en adelante.

A continuación sobre arrastre sobre el panel el control NoticiasPortada.ascx previamente diseñado, que albergará el contenido principal de la página Default.

Finalmente añada un HyperLink (hl_veranteriores) debajo del control con las siguientes propiedades:

```
<asp:HyperLink ID="hl_veranteriores" runat="server" CssClass="etiquetas"
NavigateUrl="~/Controles/Noticias/BuscaNoticias.aspx" SkinID="titulosmaster">Ver
noticias anteriores</asp:HyperLink>
```

Mediante este control se accederá a la página BuscaNoticias.aspx desde donde se podrán consultar otras noticias publicadas anteriormente.

Para finalizar con la página Default.aspx y adelantando un poco la aplicación de estilo, haga que la dicha página se derive de una clase denominada BasePage y no de

System.Web.UI.Page que es la clase base por defecto, cree dentro de la carpeta App_Code la clase BasePage.cs y por el momento déjela sin editar, más adelante se regresará a este punto.

```
public partial class _Default : BasePage
```

También asigné un título a esta página, para que sea el mostrado en el navegador: “Programación Orientada a la Web”, esto en la propiedad Title de la página, todo esto deberá hacerlo de aquí en adelante para todas las páginas .aspx.

6.5. Página Noticia

6.5.1. Descripción

La página específica de una noticia se dividirá principalmente en dos partes. Por un lado, y en la parte superior, se mostrará el titular, el resumen, y el cuerpo de la noticia, así como la imagen asociada a la noticia. Por otro lado, en la parte inferior, se dispone de un espacio dedicado a las opiniones sobre la noticia. En el se podrá leer todas las opiniones vertidas por los usuarios de la Web, y también se posibilitará la opción de opinar sobre la misma, si bien esta opción sólo estará habilitada para cuando el usuario se haya autenticado previamente.

Las opiniones aparecerán de la más reciente a la más antigua, y haciendo clic en los botones de la parte inferior se podrá avanzar o retroceder en la paginación de las opiniones.

6.5.2. Desarrollo

La página Noticia.aspx está formada fundamentalmente por un control de usuario llamado Noticia.ascx. En primer se describiremos la creación de dicho control para acto seguido pasar a detallar el desarrollo de la página en sí.

6.5.2.1. Control Noticia

23/06/2008

UNA OLEADA DE CORREOS FRAUDULENTOS INTENTA OBTENER CLAVES DE USUARIOS DE TELEFÓNICA

Con el objetivo de obtener las claves privadas de los usuarios de Telefónica, lo que se conoce como "phishing", se está produciendo un envío masivo de correos fraudulentos que simulan proceder de la citada compañía, según denunció hoy la Asociación de Internautas (AI).

Después de "analizar este nuevo intento de pesca de incautos podemos ver con claridad su mala elaboración, y donde el ciberdelincuente roza el ridículo por su mala traducción y expresiones usadas en el correo trampa enviado a las posibles victimas de esta técnica de ingeniería social", indicaron desde AI. Así pues, se espera que "el número de victimas de este engaño sea mínimo, aunque no se puede descartar que algún cliente despistado pique en la trampa", apuntó la asociación.

OPINIONES

Existen 3 opiniones vertidas por los usuarios acerca de esta noticia

magda 25/06/2008 11:48:00 a.m.

Por eso no es recomendable tener la misma contraseña, para todas nuestras cuentas de correo electrónico, banco, etc.

ariel 24/06/2008 11:47:00 a.m.

Se debe tener mucho cuidado con nuestras contraseñas

carol 23/06/2008 11:47:00 a.m.

Cuidado amigos con los SPAM y con correos de usuarios que no conocemos

<< First | < Prev | Next > | Last >>

Tu opinión - Sólo los usuarios registrados pueden enviar opiniones

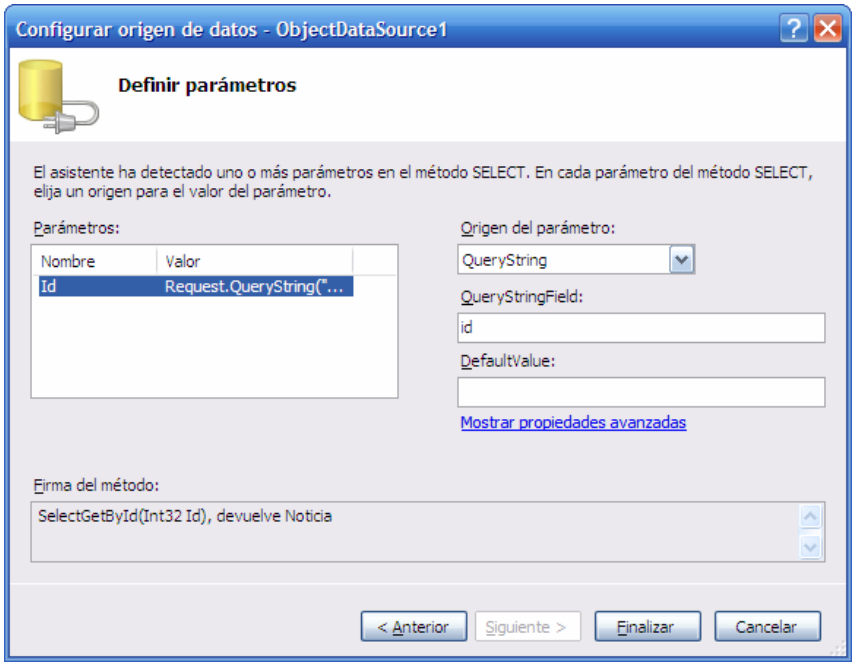
Enviar opinión

Fig. 73 Página Noticia

Dentro de este control se puede distinguir dos partes principales: la información referente a la noticia y las opiniones asociadas con la misma. Agregue un control de usuario dentro de la carpeta ~/Controles/Noticias denominado Noticia.ascx, a continuación se explica cada una de las dos partes que conforman dicho control.

Información de la noticia

Para mostrar la información de la noticia se va a hacer uso de un control Repeater enlazado a un ObjectDataSource. En primer lugar agregue este ObjectDataSource (ObjectaDataSource1) y los configúrelo para que el método asociado con la operación SELECT sea SelectGetById de la clase BLLNoticia. Este método devuelve un objeto de la clase Noticia y recibe como parámetro un entero que corresponde con el identificador de dicha noticia. Después de definir el método de datos tendremos que elegir un origen para el valor del parámetro. Este origen será una QueryString denominada id.



Una vez configurado el ObjectDataSource proceda a añadir el Repeater (Repeater1) y a elegir como origen de datos del mismo dicho ObjectDataSource.

Edite el control Repeater de la siguiente manera:

```
<asp:Repeater ID="Repeater1" runat="server" DataSourceID="ObjectDataSource1">
   <ItemTemplate>
      <table width="100%" cellpadding="12px">
         <tr>
            <td>
               <asp:Label ID="etFecha" runat="server" Text='<%#Eval("_Fecha","{0:d}") %>'
CssClass="etiquetaspequeñas">
               </asp:Label>
               <br />
                  <asp:Label ID="et_Titular" runat="server" Text='<%# Eval("_Título") %>'
CssClass="titulos"
                  ForeColor="DarkBlue" SkinID="titulosmaster"> </asp:Label>
               <br />
               <br />
                  <asp:Label ID="etResumen" runat="server" Text='<%#
Eval("_Resumen")%>' CssClass="resumenes"></asp:Label>
            </td>
         </tr>

         <tr>
            <td>
               <asp:Image runat="server" ID="imgNoticia" CssClass="imgnoticia"
ImageAlign="right"
                  ImageUrl='<%# Eval("_Imagen", "{0}") %>' BorderStyle="Solid"
BorderWidth="2px"
                  BorderColor="DarkGray" GenerateEmptyAlternateText="true"
AlternateText="Imagen no disponible" /><div
                     class="textos">
                     <%# Eval("_Cuerpo").ToString()%>
                     <br/>
                  </div>
            </td>
         </tr>
</ItemTemplate>
<FooterTemplate>
</table>
</FooterTemplate>
</asp:Repeater>
```

En el código en la plantilla ItemTemplate del Repeater, se sitúa una tabla formada por dos filas y una columna. En la primera fila se ubican tres etiquetas, etFecha, et_Titular y etResumen, cuyas propiedades Text se encuentran enlazadas con los campos de datos _Fecha, _Títular y _Resumen respectivamente. Por otro lado, en la segunda fila se coloca una imagen cuya propiedad ImageUrl está enlazada con el campo de datos _Imagen, y también una capa en cuyo interior se muestra el texto enlazado con el campo _Cuerpo. De esta manera, el texto rodeará a la imagen, cosa que no sucedería si se utilizara un control Label. Como podemos

observar, se aplican diferentes clases CSS a los elementos para dotarlos de un estilo en concreto.

Hay que destacar que el cierre de la tabla se encuentra en el interior de la plantilla FooterTemplate del Repeater.

Opiniones asociadas a la noticia

Para mostrar las opiniones asociadas a la noticia se hace uso de un control DataList cuyo contenido irá paginado.

Antes de continuar con la edición de las opiniones, agregue al fichero Noticia.ascx, un separador y una etiqueta “Opiniones”, para que cuando el usuario pulse sobre las opiniones de una noticia se le redirigirá a esta sección de la página (vea la sección ControlNoticiasPortada).

```
.....
.....

</asp:Repeater>
<hr />
<a id="Opiniones"></a>
```

En vista diseño añada un control UpdatePanel (UpdatePanel1), dentro de este control añada otro objeto ObjectDataSource (ObjectDataSource2), elija como objeto comercial para recuperar los datos la clase BLLOpinión y como método de dicha clase para asociado con la operación SELECT el método ListByIdNoticiasPaginadas.

Este método devuelve una lista de opiniones (ListOpinión) y recibe tres parámetros de entrada, el parámetro id, cuyo origen será una QueryString denominada id (es decir, la misma que en el ObjectDataSource1, y los parámetros pageIndex y pageSize. Estos dos parámetros se usan para definir los elementos a mostrar.

La sintaxis declarativa de este objeto quedará de la siguiente manera:

```
<asp:ObjectDataSource ID="ObjectDataSource2" runat="server"
DataObjectTypeName="PortalWeb.BO.Opinión"
DeleteMethod="Delete" OldValuesParameterFormatString="original_{0}"
SelectMethod="ListByIdNoticiasPaginadas"
```

```
TypeName="PortalWeb.BLL.BLLOpinión" UpdateMethod="AddUpdate"
OnSelecting="ObjectDataSource2_Selecting">
   <SelectParameters>
      <asp:QueryStringParameter Name="id" QueryStringField="id" Type="Int32" />
      <asp:Parameter Name="pageIndex" Type="Int32" />
      <asp:Parameter Name="pageSize" Type="Int32" />
   </SelectParameters>
</asp:ObjectDataSource>
```

A continuación en la parte superior del control UpdatePanel1 una tabla con una columna y tres filas. En la primera fila situé dos etiquetas, la primera tendrá el texto “Opiniones” (SkinId = titulosmaster, CssClass=titulospequeños), y la segunda irá destinada a mostrar un mensaje informando del número de opiniones existentes (etNumOpiniones, CssClass = textosnegritos).

```
.titulospequeños
{
        font-size: small;
        text-transform: uppercase;
        color: black;
        font-family: Tahoma;
        font-variant: small-caps;
        font-weight: bold;
        text-align: left;
}

.textosnegrita
{
        color: black;
        font-family: Tahoma;
        text-align: justify;
        font-size: small;
        font-weight: bold;
}
```

En la segunda fila sitúe un control DataList (DataList1), que irá enlazado con el ObjectDataSource(ObjectDataSource2) previamente configurado. Y en la tercera fila sitúe los cuatro botones que constituirán la interfaz de paginación. (“btFirs”, “btPrev”, “btNex” y “btLast”).

Seguidamente edite la plantilla ItemTemplate del control DataList. En ella ubique una dos etiquetas, lb_usuario (CssClass = textos, SkinID = titulosmaster) y lb_fecha (CssClass = textospeq, ForeColor = Gray). La propiedad Text de cada una irá enlazada con el campo de datos _Usuario y el campo de datos _Fecha, respectivamente. Coloque un Enter y a continuación otra etiqueta, lb_texto, cuya propiedad Text irá enlazada con el campo de datos _Texto (CssClass = textos).

```
.textospeq
{
        font-family: Tahoma;
        font-size: x-small;
}
```

Por otro lado, editaremos la plantilla SeparatorTemplate, en la que situará un tag <HR>, es decir una regla horizontal. Ésta servirá para distinguir de manera más clara un comentario de otro.

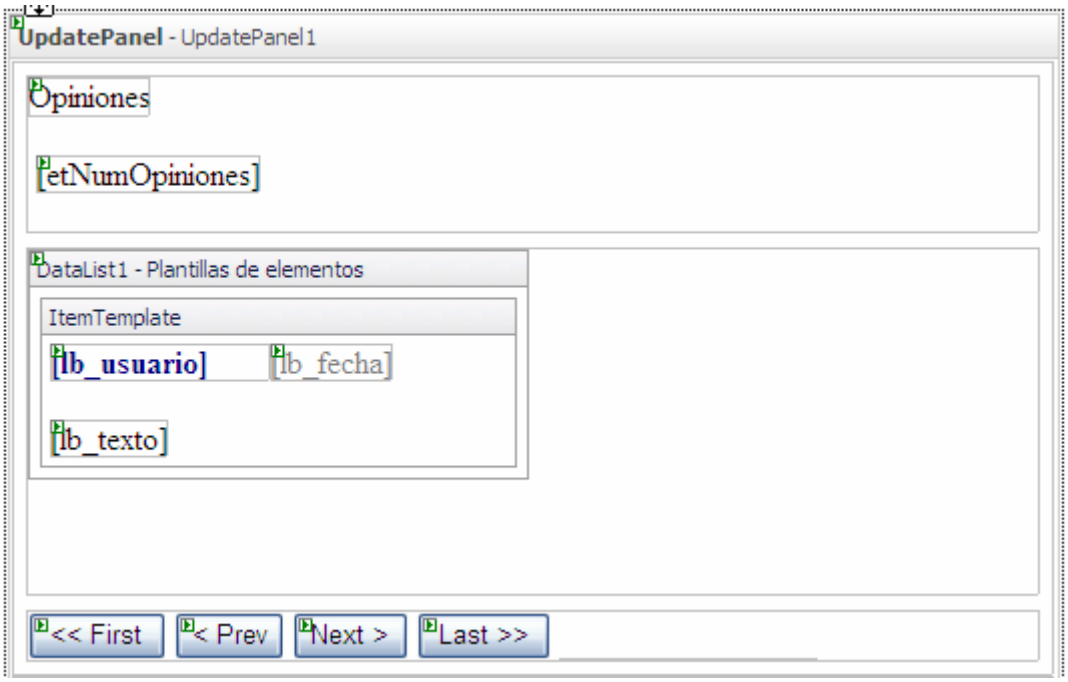


Fig. 74 Sección Opiniones.

Como se ha mencionado anteriormente, se va a recurrir al uso de métodos paginados, para ello, en primer lugar, agregue (archivo Noticia.ascx.cs) las propiedades StartRowIndex, MaximumRows y TotalRowCount. Las dos primeras almacenarán el estado de la vista del control, de forma que irán variando dependiendo de donde nos situemos. Se procede de esta forma para evitar que si está en una determinada noticia, navegando por sus opiniones, y a continuación se selecciona otra noticia de las expuestas en el control Noticias más Comentadas, los valores de estas propiedades no varíen y sean erróneos.

```
private int StartRowIndex
   {
      get
      {
         string key = "StartRowIndex" + noticiaid.ToString();
         object o = ViewState[key];
         if (o == null)
```

```
            return 0;
         else
            return (int)o;
      }
      set
      {
         string key = "StartRowIndex" + noticiaid.ToString();
         ViewState[key] = value;
      }
   }

   private int MaximumRows
   {
      get
      {
         string key = "MaximumRows" + noticiaid.ToString();
         object o = ViewState[key];
         if (o == null)
            return 5;
         else
            return (int)o;
      }
      set
      {
         string key = "MaximumRows" + noticiaid.ToString();
         ViewState[key] = value;
      }
   }

   private int TotalRowCount
   {
      get
      {
         return BLLOpinión.NumOpinionesByIdNoticia(noticiaid);
      }
   }
```

A continuación añada el manejador para el evento Selecting del ObjectDataSOurce2, cuyo código será el siguiente:

```
protected void ObjectDataSource2_Selecting(object sender,
ObjectDataSourceSelectingEventArgs e)
   {
 e.InputParameters["pageIndex"] = StartRowIndex;
 e.InputParameters["pageSize"] = MaximumRows;
….}
```

Cree los manejadores para el evento Clic de los botones de la interfaz de paginación:

```
   protected void btFirst_Click(object sender, EventArgs e)
   {
```

```
        // Return to StartRowIndex of 0 and rebind data
        StartRowIndex  = 0;
        DataList1.DataBind();
    }
    protected void btPrev_Click(object sender, EventArgs e)
    {
        // Subtract MaximumRows from StartRowIndex and rebind data
        StartRowIndex -= MaximumRows;
        DataList1.DataBind();
    }
    protected void btNext_Click(object sender, EventArgs e)
    {
        // Add MaximumRows to StartRowIndex and rebind data
        StartRowIndex += MaximumRows;
        DataList1.DataBind();
    }
    protected void btLast_Click(object sender, EventArgs e)
    {
        //Set StartRowIndex to last page's starting row index and rebind data
        StartRowIndex = ((TotalRowCount - 1) / MaximumRows) * MaximumRows;
        DataList1.DataBind();
    }
```

Por otro lado, para que la etiqueta destinada a mostrar información acerca del número de opiniones existentes (etNumOpiniones) realice su función, añada el siguiente código al manejador del evento Selecting del objeto ObjectDataSource2.

```
etNumOpiniones.Text = GetNOpiniones();
```

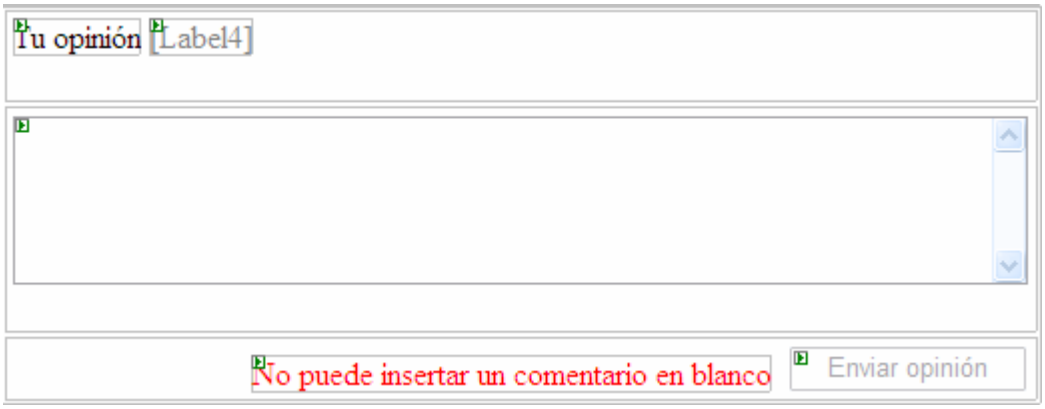
El código siguiente corresponde al método GetOpiniones()

```
protected string GetNOpiniones()
    {
        int opiniones = TotalRowCount;
        if (opiniones == 0)
        {
              return ("Todavía no hay opiniones sobre esta noticia. ");
        }
         else if (opiniones > 1)
        {
            return ("Existen " + opiniones.ToString() + " opiniones vertidas" +
                  " por los usuarios acerca de esta noticia");
         }
       else
        {
            return ("Existe " + opiniones.ToString() + " opinión vertida por " +
                  "los usuarios acerca de esta noticia");
        }
}
```

Si no hay opiniones se deberían desactivar todos los botones de la interfaz de paginación, complete con el siguiente código el manejador del evento Selecting del ObjectDataSource2.

```
// Desactiva los botones de la interfaz de paginación, si es necesario.
        btFirst.Enabled = StartRowIndex != 0;
        btPrev.Enabled = StartRowIndex != 0;

        int LastPageStartRowIndex = ((TotalRowCount - 1) / MaximumRows) *
MaximumRows;
        btNext.Enabled = StartRowIndex < LastPageStartRowIndex;
        btLast.Enabled = StartRowIndex < LastPageStartRowIndex;
if (etNumOpiniones.Text == "Todavía no hay opiniones sobre esta noticia. ")
        {
            btFirst.Visible = btLast.Visible = btPrev.Visible =
            btNext.Visible = false;
}
```

Por último debe añadir los elementos necesarios para insertar nuevas opiniones. Para ello: en primer lugar añada en el UpdatePanel1, una tabla con una columna y tres filas. En la primera fila coloque dos etiquetas, una para indicar que se puede añadir opiniones (CssClass=etiquetas), y la otra destinada a mostrar mensajes de información que indicarán que no es posible añadir opiniones cuando los usuarios no hayan iniciado sesión (Label4, CssClass=textos). En la segunda fila irá situada una caja de texto (CssClass=textos), donde el usuario introducirá el correspondiente comentario. Finalmente, y ya en la última fila, coloque un control de validación RequiredFieldValidator, que irá asociado con la caja de texto, y un botón para validar la operación.



Tanto la caja de texto, como el control de validación y el botón, formarán parte de un mismo grupo de validación (“addopinion”), de tal manera que cuando se haga clic en el botón se

produzca dicha validación, tome esto en cuanta para la validación de todos los controles que aún faltan en el desarrollo del Portal.

Para insertar nuevas opiniones, los usuarios deberán haber iniciado sesión o de no ser así, el botón Button1 estará desactivado. Para que esto suceda, añada el siguiente código al manejador del evento Load de la página (Page_Load).

```
protected void Page_Load(object sender, EventArgs e)
    {
        noticiaid = Convert.ToInt32(Request.QueryString["id"]);
        Noticia minoticia =   BLLNoticia.SelectGetById(noticiaid);
        titulo = minoticia._Título;
        if (!HttpContext.Current.User.Identity.IsAuthenticated)
            Label4.Text = " - Sólo los usuarios registrados pueden enviar opiniones";
        else
            TextBox1.Enabled = Button1.Enabled = true;
        Page.Title = titulo;
    }
```

Por otro lado el botón Button1, tiene asociado el siguiente controlador:

```
protected void Insertar_Opinion(object sender, EventArgs e)
    {
        Opinión miopinion = new Opinión();
        miopinion._Comentario = TextBox1.Text;
        miopinion._Id_Noticia = Convert.ToInt32(Request.QueryString["id"]);
        miopinion._Usuario = HttpContext.Current.User.Identity.Name.ToString();
        miopinion._Fecha = DateTime.Now;
        BLLOpinión.AddUpdate(miopinion);
        TextBox1.Text = " ";
        StartRowIndex = 0;
        DataList1.DataBind();
    }
```

Este método crea un nuevo objeto de la clase Opinión y rellena sus propiedades a partir del contenido de la caja de la caja de texto, del valor de la QueryString (que corresponde con el identificador de la noticia), del usuario actual y de la fecha del sistema. Después de guardar la opinión, borra el contenido de la caja de texto, establece a cero la propiedad StartRowIndex y llama al método DataBind del DataList.

6.5.2.2. Página Noticia.aspx

En esta página, y al igual que ocurría en el caso de Default.aspx, sitúe un panel en cuyo interior se ubica el control Noticia.ascx previamente diseñado. A continuación añada un extender RoundedCorner para lograr que el panel presente esquinas redondeadas. Esta página deberá estar situada en la carpeta Noticias.

6.6. Página Buscador de Noticias

6.6.1. Descripción

Cuando se accede al portal, en la portada, tan sólo se muestran las noticias más recientes. Pero si se desea buscar alguna noticia no tan reciente, se debe hacer uso del buscador de noticias. Diseñado a modo de hemeroteca, permitirá averiguar en qué días se han publicado noticias, a través de una interfaz en forma de calendario. Haciendo clic sobre el mes deseado, aparecerán marcados en rojo aquellos días para los que existen noticias. Y si se pulsa sobre cualquiera de ellos, se remitirá a una página donde se verán todas las noticias de ese día, dispuestas de forma ordenada, como en la portada. También se tiene la opción de hacer búsquedas de noticias, eligiendo un día exacto mediante unas listas desplegables.

Fig. 75 Buscador de Noticias.

6.6.2. Desarrollo

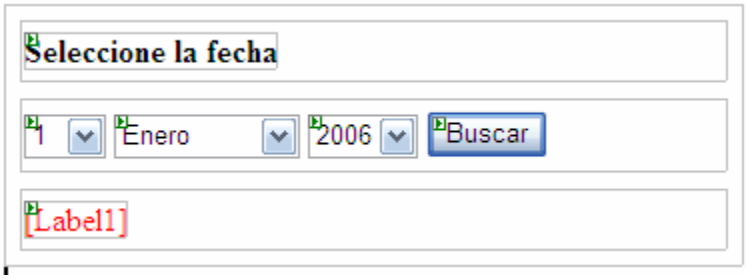
Para la realización de esta página, denominada BuscaNoticias.aspx, se desarrollan dos controles de usuario, que posteriormente pasarán a formar parte de la misma (Ubicados todos en la carpeta Noticias). El primero se denominará CalendarioNoticias, y contendrá el calendario donde se visualizarán las fechas en las que se han publicado noticias. El segundo control, denominado BuscaNoticia.ascx contendrá los desplegables para la realización de búsquedas.

6.6.2.1. Control CalendarioNoticias.ascx

Este control está formado por un Calendar que se configura de manera similar al utilizado en el control Calendario de eventos situado en la página maestra. Lo único que cambiará será el aspecto del mismo y que en lugar de obtener las fechas a señalar mediante el método BLLEvento.GetFechasFuturas() lo hará con el método BLLNoticia.Select.GetFechas(). Cuando se pinche sobre una de las fechas destacadas, se remitirá a la página NoticiasFecha.aspx pasando como parámetro QueryString la fecha seleccionada.

6.6.2.2. Control BuscaNoticia.ascx

Para la realización de este control, agregue en primer lugar, una tabla compuesta de una columna y tres filas. El la primera fila, sitúe una etiqueta (CssClass = etiquetas, SKinID = titulosmaster) con el texto “Selecciona la fecha”. En la segunda, un botón y tres controles del tipo DropDownList: uno para seleccionar el día, otro para el mes y otro para el año. Y en la última fila coloque una etiqueta destinada a mostrar un mensaje en caso de que no haya noticias del día elegido.



El botón tiene asociado un manejador para su evento clic:

```
protected void btBuscarNoticias_Click(object sender, EventArgs e)
    {
        string dia = listDia.SelectedValue;
        string mes = listMes.SelectedValue;
        string año = listAño.SelectedValue;
        string fecha = dia + "/" + mes + "/" + año;
        string[] arrayfechas = BLLNoticia.SelectFechas();
        int contador = arrayfechas.Length;
        bool ok = false;
        int i = 0;
        while (i < contador)
        {
            if (fecha == arrayfechas[i])
                ok = true;
            i++;
            if (ok)
                Response.Redirect("~/Controles/Noticias/NoticiasFecha.aspx?fecha="
                + fecha);
            else
                Label1.Text = "No hay Noticias del día " + fecha;
        }
    }
```

En primer lugar se almacenan los valores seleccionados en los tres DropDownList concatenándolos para originar una cadena con el formato de fecha. Seguidamente, en un array de tipos String se guarda el resultado de llamar a la función BLLNoticia.SelectFechas(), es decir cada una de las fechas en las que se han publicado noticias. Acto seguido, mediante un bucle, se recorre dicho array comparando cada uno de sus elementos con la fecha seleccionada mediante los DropDownList. Si se encuentra la fecha, se redirige la página NoticiasFecha.aspx, pasándole como parámetro dicha fecha. Y en caso contrario, la etiqueta destinada para ello indica que no existen noticias con esa fecha de publicación.

6.6.2.3. Página BuscaNoticias.aspx

Finalmente, una vez que diseñado los dos controles, se crea la página BuscaNoticias.aspx. Coloque Colocaremos en esta un UpdatePanel, dentro este un Panel y en su interior ambos controles. Recuerde establecer las esquinas redondeadas del Panel.

6.7. Página Noticias Fecha

6.7.1. Descripción

Esta página mostrará las noticias publicadas en una determinada fecha, tras haber seleccionado la misma en la página BuscaNoticias.aspx. Su apariencia será similar a la portada del sitio Web, sólo que en lugar de tener por título la etiqueta “Últimas Noticias” tendrá una con el texto “Noticias del día DD/MM/YYYY”.

6.7.2. Desarrollo

Para el desarrollo de esta página de utilizará un único control denominado NoticiasFechas.ascx.

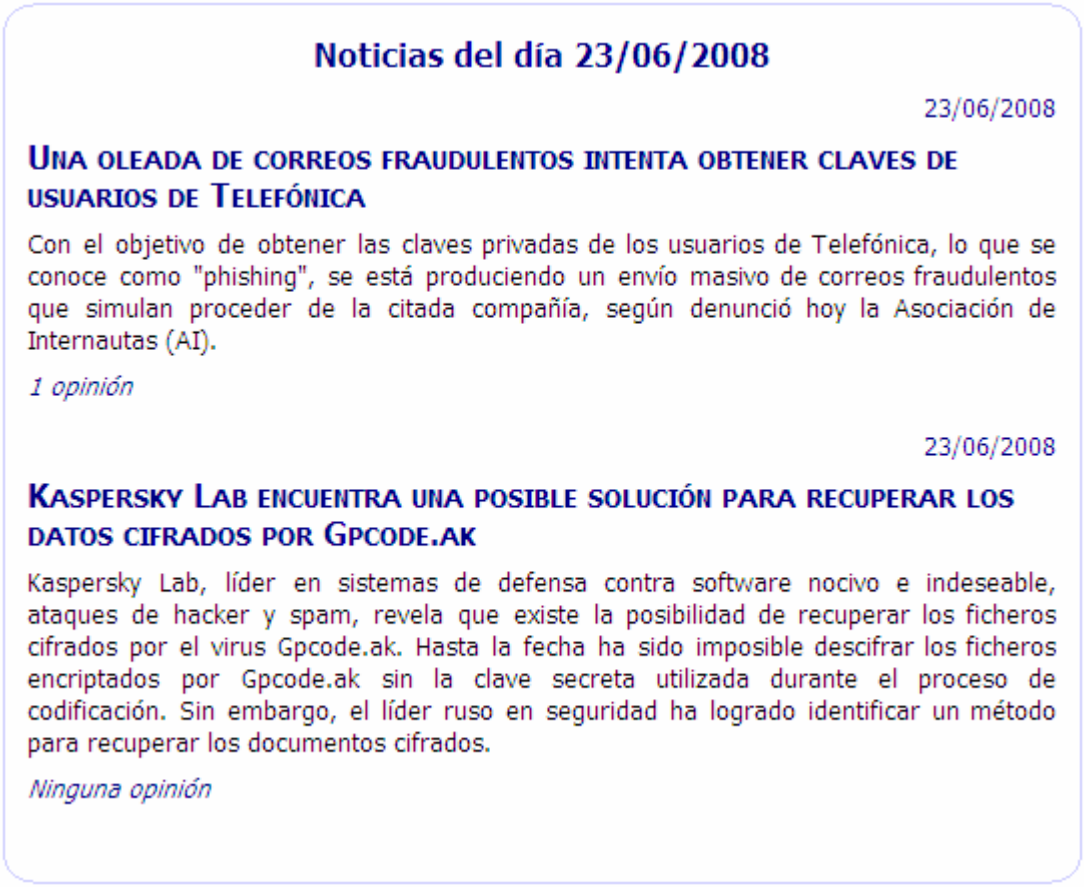


Fig. 76 Noticias por Fecha.

6.7.2.1. Control NoticiasFechas.ascx

Para mostrar los datos utilice un control de enlace a datos del tipo DataList que estará enlazado a un ObjectDataSource. Primeramente añada dicho ObjectDataSource y elija el objeto comercial BLLNoticia, y como método del objeto comercial asociado a la operación SELECT el método ListByFechas. Este método recibe como parámetro una cadena de texto correspondiente con la fecha de publicación de las noticias a mostrar. El origen de este parámetro será una QueryString denominada fecha. A continuación añada el DataList y seleccione como origen de datos el ObjectDataSource. Dicho DataList será exactamente igual que el situado en la página Default.aspx, recuerde también aplicar los estilos. Lo único que variará será la plantilla de cabecera. En este caso, estará formada únicamente por una Label cuya propiedad Text vendrá definida por el método GetTitulo().

El código de este método es:

```
protected string GetTitulo()
    {
        return "Noticias del día " + Request.QueryString["fecha"].ToString();
    }
```

6.7.2.2. Página NoticiasFecha.aspx

Finalmente, una vez que tenga listo el control, cree la página NoticiasFecha.aspx y proceda a insertarlo en la misma, recuerde insertarlo dentro de un Panel y aplicarle esquinas redondeadas.

6.8. Página RSS20.aspx

6.8.1. Descripción

El objetivo de la creación de esta página es posibilitar la generación de nuestro propio canal RSS a partir de las noticias albergadas en la base de datos, y facilitar que otros usuarios de la red se suscriban a él. Para acceder a ella, habrá que hacer clic en el pequeño icono de RSS situado en el panel de noticias de la página principal.

6.8.2. Desarrollo

Para su puesta en práctica se creará un documento en XML, en el que todos los datos deberán ir correctamente ordenados con sus correspondientes etiquetas de principio y final según lo establecido en la normativa del propio lenguaje de marcado XML. Así se creará nuestro propio Canal que puede contener varios artículos o ítems, que serán legibles a través de diversas aplicaciones (navegadores, bases de datos, etc.)

Este será el código que contenga la página:

```
private void Page_Load(object sender, System.EventArgs e)
{
          // Cacheamos la respuesta para no estar constantemente generando el fichero
          Response.ContentType = "text/xml";
          Response.ContentEncoding = Encoding.UTF8;
          MemoryStream ms = new MemoryStream();
          XmlTextWriter writer = new XmlTextWriter(ms, Encoding.UTF8);

          //Sólo las noticias de la portada
          ListNoticia misnoticias = BLLNoticia.SelectListPortada();
          writer.WriteStartDocument();
          //Creamos nuestro canal, con los elementos estáticos
          writer.WriteStartElement("rss");
          writer.WriteAttributeString("version", "2.0");
          writer.WriteStartElement("channel");
          writer.WriteElementString("title", "Últimas noticias de Programación Orientada a la
Web");
          writer.WriteElementString("link", "http://localhost/PortalWeb/Default.a.spx");
          writer.WriteElementString("description", "Las últimas noticias de POW , el sitio de
referencia para conocer diversas noticias.");
          writer.WriteElementString("ttl", "60");

          //Agregamos los elementos dinámicos a nuestro canal
          foreach (Noticia noticia in misnoticias)
          {
             writer.WriteStartElement("item");
             writer.WriteElementString("title", noticia._Título);
            writer.WriteElementString("link",
String.Format("http://localhost/PortalWeb/Controles/Noticias/Noticia.aspx?id={0}",
noticia._Id_Noticia.ToString()));
             writer.WriteElementString("description",  noticia._Resumen );

             writer.WriteElementString("pubDate", noticia._Fecha.ToString("R"));
             writer.WriteEndElement();
          }

          //cierro channel
          writer.WriteEndElement();
```

```
        //cierro rss
        writer.WriteEndElement();
        writer.WriteEndDocument();
        writer.Flush();

        ms.Seek(0, SeekOrigin.Begin);
        Response.Write((new StreamReader(ms)).ReadToEnd());
        writer.Close();
        Response.AddCacheItemDependency("cachecontrol");
}
```

Como se puede apreciar, en primer lugar se especifica la naturaleza del contenido del fichero, así como su codificación. Acto seguido se declara un objeto MemoryStream para el almacenamiento temporal del fichero y otro del tipo XmlTextWriter para la escritura en el mismo. Después, se recoge en un objeto todas las noticias existentes en la portada y se procede a rellenar el fichero Xml con los contenidos de las mismas, situados entre etiquetas.

Para ello, y en primer lugar, se indica que la versión que a desarrollar será la 2.0. Seguidamente se abren las etiquetas del canal y se indica algunos de sus atributos, como su descripción, título o enlace.

A continuación, y para mostrar cada uno de los elementos o ítems que se publicarán en nuestro canal, se inicia un bucle foreach. Mediante dicho bucle y para cada noticia, se indica el título, enlace, descripción y fecha de publicación de cada una de las noticias, acudiendo para ello a los respectivos campos de cada objeto noticia. Todos ellos serán añadidos al objeto writer.

Una vez hecho esto, se procede a cerrar tanto las etiquetas del canal, como la de la propia rss. Finalmente mediante la función Seek se sitúa en el inicio del fichero generado para posteriormente leerlo de principio a fin y escribirlo así en la secuencia de salida de la respuesta HTTP. Hasta aquí termina la edición de todo lo relacionado con las noticias, recuerde colocar estos controles y páginas en la carpeta Noticias, y de colocar en las páginas .aspx, un control Panel para aplicarle esquinas redondeadas. Por último cree dentro de esta carpeta (Noticias) otra denominada Fotos, la cual será utilizada más adelante.

6.9. Página Eventos

6.9.1. Descripción

Fig. 77 Página Eventos.

Se dividirá principalmente en dos zonas. En la zona de la izquierda podrá encontrar un listado detallado de los próximos eventos, donde se mostrará desde una pequeña descripción del evento, hasta el enlace oficial en el que obtener más información sobre dicho acontecimiento.

Por otro lado, en el margen derecho se dispone de un buscador avanzado de eventos. Se puede realizar búsquedas por ciudad o por fecha, ésta última con la ayuda de un calendario. Tras hacer clic en cualquiera de estos elementos, se remitirá a una página específica donde podremos ver los eventos programados para esos criterios de búsqueda. Además, también en este margen, y justo debajo del buscador, podrá encontrar las crónicas de los eventos más recientes con tan sólo hacer clic en aquella que se desee.

6.9.2. Desarrollo

En este apartado se va a describir el desarrollo de las páginas Eventos.aspx y Review.aspx. Para el desarrollo de la página Eventos.aspx se emplearán tres controles de usuario que crearemos previamente: ProximosEventos.ascx, BuscaEventos.ascx y Rview.ascx.

6.9.2.1. Control ProximosEventos.ascx

Este control está destinado a mostrar el listado de los próximos eventos y está formado principalmente por un control Repeater.

Añada una tabla con una columna y tres filas, en la primera coloque una etiqueta con el texto “Próximos Eventos”, establezca el estilo igual que el encabezado de los otros controles de usuario. En la segunda fila añada un control Repeater (Repeater1) cuyo código es el siguiente:

```
<asp:Repeater ID="Repeater1" runat="server" EnableViewState="false"  >
            <ItemTemplate>
            <div style="text-align: center">
                  <table border="0" style="width: 330px">

                  <tr>
                        </br>
                        <td align="left" colspan="2" rowspan="1" style="height: 40px">
                           <asp:Label ID="etTitulo" runat="server" CssClass="textosnegrita"
ForeColor="Black"
```

```
                        Text='<%# Eval("_Título") %>' Font-Underline="True" Font-
Bold="True" SkinID="titulosmaster"></asp:Label><br />
                    </td>
                </tr>

                <tr>
                    <td rowspan="3" style="width: 100px">
                        <asp:Image ID="Image1" runat="server" ImageUrl='<%#Eval("_Foto")
%>' ImageAlign="TextTop" /></td>
                    <td style="width: 100px">
                        <asp:Label ID="etFecha" runat="server" CssClass="textos"
ForeColor="Black" Text='<%# Eval("_Fecha", "{0:d}")%>'></asp:Label></td>
                </tr>

                <tr>
                    <td style="width: 100px">
                    <asp:Label ID="etCiudad" runat="server" CssClass="textos"
ForeColor="Black" Text='<%#Eval("_Ciudad") %>'>
                    </asp:Label></td>
                </tr>
                <tr>
                    <td style="width: 100px">
                        <asp:HyperLink ID="HyperLink1" runat="server" CssClass="textos"
NavigateUrl='<%#Eval("_Enlace", "{0}") %>' Text='Web Oficial' Target="_blank">
                        </asp:HyperLink>
                    </td>
                </tr>

                <tr>
                    <td colspan="2">
                    <br />
                      <asp:Label ID="Label2" runat="server" CssClass="textos" Text='<%#
Eval("_Descripción") %>' Width="315px">
                        </asp:Label></td>
                </tr>
            </table>
        </div>

        </ItemTemplate>
</asp:Repeater>
```

Para obtener los datos de forma paginada, se recurre al siguiente método, que acudirá al método paginado correspondiente de la clase BLLEvento de forma similar a como ocurría con la paginación de otras secciones como la de los comentarios a noticias.

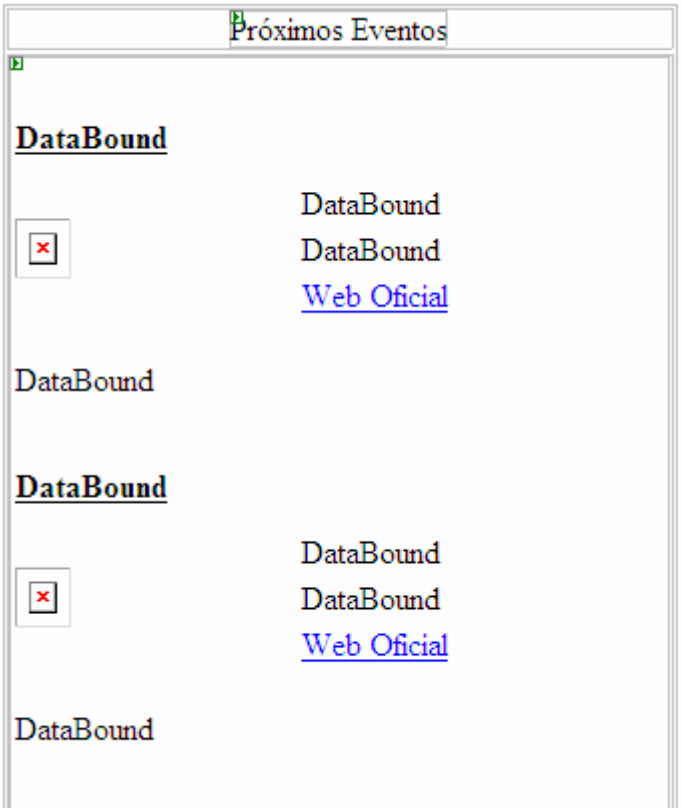


Fig. 78 Diseño de la página Eventos.

```
public partial class Controles_Eventos_ProximosEventos : System.Web.UI.UserControl
{

    ListEvento misEventos;
    string fecha;
    string descripcion;
    string ciudad;
    int numpag = 1;
    int pagtot = 1;
```

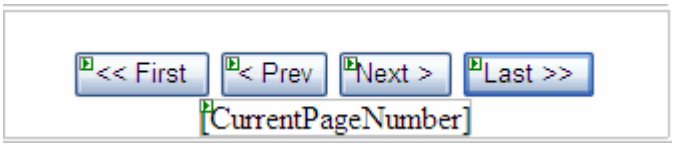
Por un lado se tiene se tiene el manejador del evento Load del control Repeater

```
protected void Page_Load(object sender, EventArgs e)
    {
        fecha = Request.QueryString.Get("fecha");
        descripcion = Request.QueryString["descripcion"];
        ciudad = Request.QueryString["ciudad"];
        CargarDatos();
    }
```

Y por otro el método CargarDatos()

```
private void CargarDatos()
   {
      if (fecha != null)
         misEventos = BLLEvento.GetEventosByFechaPaginados(fecha, StartRowIndex,
MaximumRows);
      else if (ciudad != null)
         misEventos = BLLEvento.GetEventosFuturosByCiudadPaginados(ciudad,
StartRowIndex, MaximumRows);
      else
         misEventos = BLLEvento.GetEventosFuturosPaginados(StartRowIndex,
MaximumRows);
      Select();
      this.Repeater1.DataSource = misEventos;
      this.Repeater1.DataBind();
   }
```

En la tercera fila de la tabla, coloque los botones de paginación y una etiqueta.



Define el método Select de la siguiente manera.

```
private void Select()
   {
      btFirst.Enabled = StartRowIndex != 0;
      btPrev.Enabled = StartRowIndex != 0;

      int LastPageStartRowIndex = ((TotalRowCount - 1) / MaximumRows) *
MaximumRows;
      btNext.Enabled = StartRowIndex < LastPageStartRowIndex;
      btLast.Enabled = StartRowIndex < LastPageStartRowIndex;
      if (TotalRowCount < MaximumRows || TotalRowCount == MaximumRows)
         pagtot = 1;
      else if ((TotalRowCount % MaximumRows) == 0)
         pagtot = (TotalRowCount / MaximumRows);
      else
         pagtot = (TotalRowCount / MaximumRows) + 1;
      CurrentPageNumber.Text = numpag.ToString() + " de " + Convert.ToString(pagtot);
   }
```

Finalmente define las propiedades StartRowIndex, MaximunRows, TotalRowCount y los controladores del evento Clic de cada uno de los botones de paginación.

```
protected void btFirst_Click(object sender, EventArgs e)
   {
     StartRowIndex = 0;
      numpag = 1;
      CargarDatos();
   }

   protected void btPrev_Click(object sender, EventArgs e)
   {
     StartRowIndex -= MaximumRows;
      if (StartRowIndex == 0)
         numpag = 1;
      else
      {
         if (MaximumRows > StartRowIndex)
            numpag = (MaximumRows / StartRowIndex) + 1;
         else
            numpag = (StartRowIndex / MaximumRows) + 1;
      }
      CargarDatos();
   }

   protected void btNext_Click(object sender, EventArgs e)
   {
      StartRowIndex += MaximumRows;
      if (MaximumRows > StartRowIndex)
         numpag = (MaximumRows / StartRowIndex) + 1;
      else
         numpag = (StartRowIndex / MaximumRows) + 1;
      CargarDatos();
   }

   protected void btLast_Click(object sender, EventArgs e)
   {
     StartRowIndex = ((TotalRowCount - 1) / MaximumRows) * MaximumRows;
      if ((TotalRowCount % MaximumRows) == 0)
         numpag = TotalRowCount / MaximumRows;
      else
         numpag = (TotalRowCount / MaximumRows) + 1;
      CargarDatos();
   }

   private int StartRowIndex
   {
      get
      {
         object o = ViewState["StartRowIndex"];
         if (o == null)
            return 0;
         else
            return (int)o;
      }
      set
      {
         ViewState["StartRowIndex"] = value;
      }
```

```
}

private int MaximumRows
{
    get
    {
        object o = ViewState["MaximumRows"];
        if (o == null)
            return 5;
        else
            return (int)o;
    }
    set
    {
        ViewState["MaximumRows"] = value;
    }
}

private int TotalRowCount
{
    get
    {
        if (fecha != null)
            return BLLEvento.GetNumEventosFuturosByFechaPaginados(fecha);
        else if (ciudad != null)
            return  BLLEvento.GetNumEventosFuturosByCiudadPaginados(ciudad);
        else
            return BLLEvento.GetNumEventosFuturosPaginados();
    }
}
```

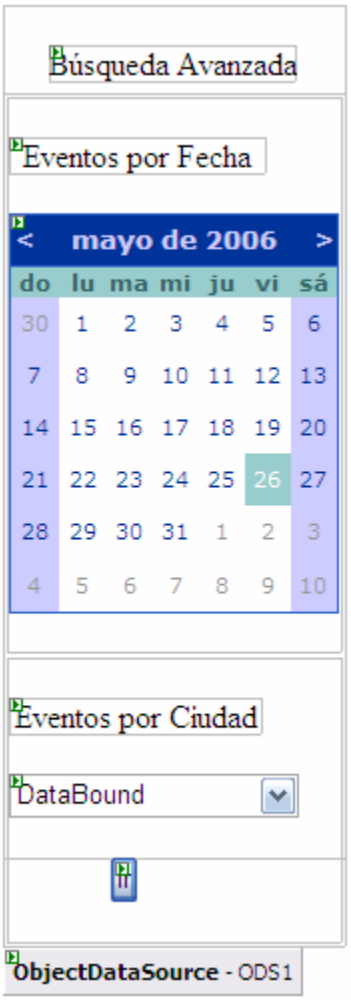
6.9.2.2. Control BuscaEventos.ascx

Ahora diseñe el control de búsqueda avanzada ubicado en el margen derecho. Éste estará formando principalmente por una tabla de una única columna y cuatro filas. En la primera de ellas ubicaremos una Label con el título del Búsqueda Avanzada, recuerde seguir aplicando los estilos.

En la segunda inserte un control de tipo Calendar acompañado por su correspondiente etiqueta. Éste calendario funcionará de manera similar al existente en la página maestra. Será capaz de recoger y marcar en los días señalados, aquellas fechas en las que se producirán Eventos. Con tan sólo pulsar en el día marcado, se redirigirá a una página en la que se mostrarán todos los Eventos programados para esa fecha.

No se trata de una página nueva, sino de la misma que la anterior, con la excepción de que en este caso solo se mostrarán aquellos eventos que coincidan con los resultados de la búsqueda pasados por QueryString. Se seguirán conservando pues los controles de la columna lateral derecha. Por tanto siga todos los pasos que ya realizó en el control Calendario Eventos.

En la tercera fila se encontrará el buscador por ciudad. Éste estará compuesto por una etiqueta y por un control de tipo DropDownList (dlistCiudad), que al desplegarse será capaz de mostrar todas las ciudades en las que van a acontecer evento. Para ello, se enlaza con el ObjectDataSource(ODS1), este recurrirá al método GetCiudadesFuturas de la clase BLLEvento.



Tras hacer clic en el botón “Ir” se redirigirá a la página específica donde se mostrarán todos los Eventos existentes en esa ciudad.

```
protected void btIrCiudad_Click(object sender, EventArgs e)
   {
      string ciudad = dlistCiudad.SelectedValue;
      Response.Redirect("~/Controles/Eventos/Eventos.aspx?Ciudad="
+ ciudad);
   }
```

6.9.2.3. Control Review.ascx

Por último, en la parte inferior de la columna derecha se situará el control dedicado a las reviews. Añada una tabla con dos filas y una columna, la primera fila te estará formada por un control DataList enlazado mediante un ObjectDataSource (odsEvento) al método GetEventosPasadosPaginados, que devolverá un listado de Eventos (ListEvento). Cuyo código es el siguiente:

```
<asp:ObjectDataSource ID="odsEvento" runat="server"
DataObjectTypeName="PortalWeb.BO.Evento"
   DeleteMethod="Delete" OldValuesParameterFormatString="original_{0}"
OnSelecting="odsEvento_Selecting"
   SelectMethod="GetEventosPasadosPaginados"
TypeName="PortalWeb.BLL.BLLEvento"
   UpdateMethod="AddUpdate">
   <SelectParameters>
      <asp:Parameter Name="startRowIndex" Type="Int32" />
```

```
        <asp:Parameter Name="maximumRows" Type="Int32" />
    </SelectParameters>
</asp:ObjectDataSource>
```

La plantilla ItemTemplate muestra el siguiente aspecto.

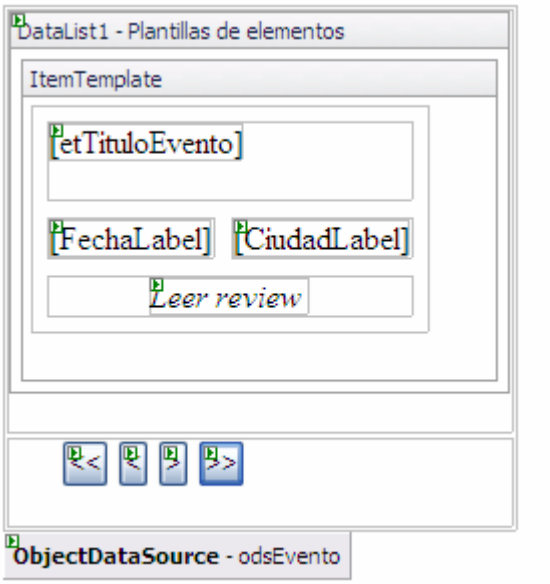


Fig. 79 Diseño del control Review.

En ella se ha insertado una tabla, con un amplio espaciamiento entre sus celdas, y que estarán destinadas a dar cobijo a diferentes elementos. Por un lado, y en la parte superior, en Título del evento, y justo debajo, la fecha y la ciudad en la que aconteció el Evento. Por último, un hipervínculo al enlace de la crónica. También cabe mencionar que el listado de las reviews estará también paginado. Por lo que debe enlazar la propiedad Text con el campo correspondiente y en el hipervínculo enlace su propiedad NavigateUrl con el campo _Id_Evento y el siguiente formato “~/Controles/Eventos/Review.aspx?id={0}”.En la plantilla HeaderTemplate coloque una etiqueta con el texto “Evento anteriores”.

En la segunda fila (**Fig. 79**), añada los controles de paginación, declare las propiedad MaximumRows, TotalRowCount y StartRowindex, igual que en control ProximosEventos.ascx, con la salvedad TotalRowCount.

```
private int TotalRowCount
  {
    get
    {
      return BLLEvento.GetNumEventosPasadosPaginados();
    }
  }
```

Defina también los controladores de los botones de paginación, siguiendo como ejemplo los definidos en el control Noticia.ascx, utilice este mismo control para definir el controlador del evento Selecting del control odsEvento.

6.9.2.4. Control Eventos.aspx

Una vez creados los tres controles anteriormente descrito proceda a la creación de la página Eventos.aspx. Para la disposición en la misma de dichos controles, cree una tabla con dos filas y dos columnas, combinando las dos celdas de la primera columna. Acto seguido insertaremos en las celdas cada uno de los controles. Finalmente, añadiremos un UpdatePanel, dentro de este un Panel que envolverá a la tabla y al que se le aplicarán las esquinad redondeadas.

6.9.2.5. Página Review.aspx

La página Reviews.aspx dispondrá de un control Repeater, muy similar al descrito anteriormente en el control ProximosEventos.ascx y que contará con todos los campos mostrados en éste último, a excepción de la descripción, que será sustituida por la propia crónica del Evento (Campo _Review). Para ello se utiliza un control ObjectDataSource, cuyo objeto comercial asociado es BLLEvento y el método correspondiente a la operación Select es SelectGetById, cuyo origen del campo id será un QueryString (id).

En esta ocasión no se contarán con los controles de la columna de la derecha. La apariencia final deberá ser:

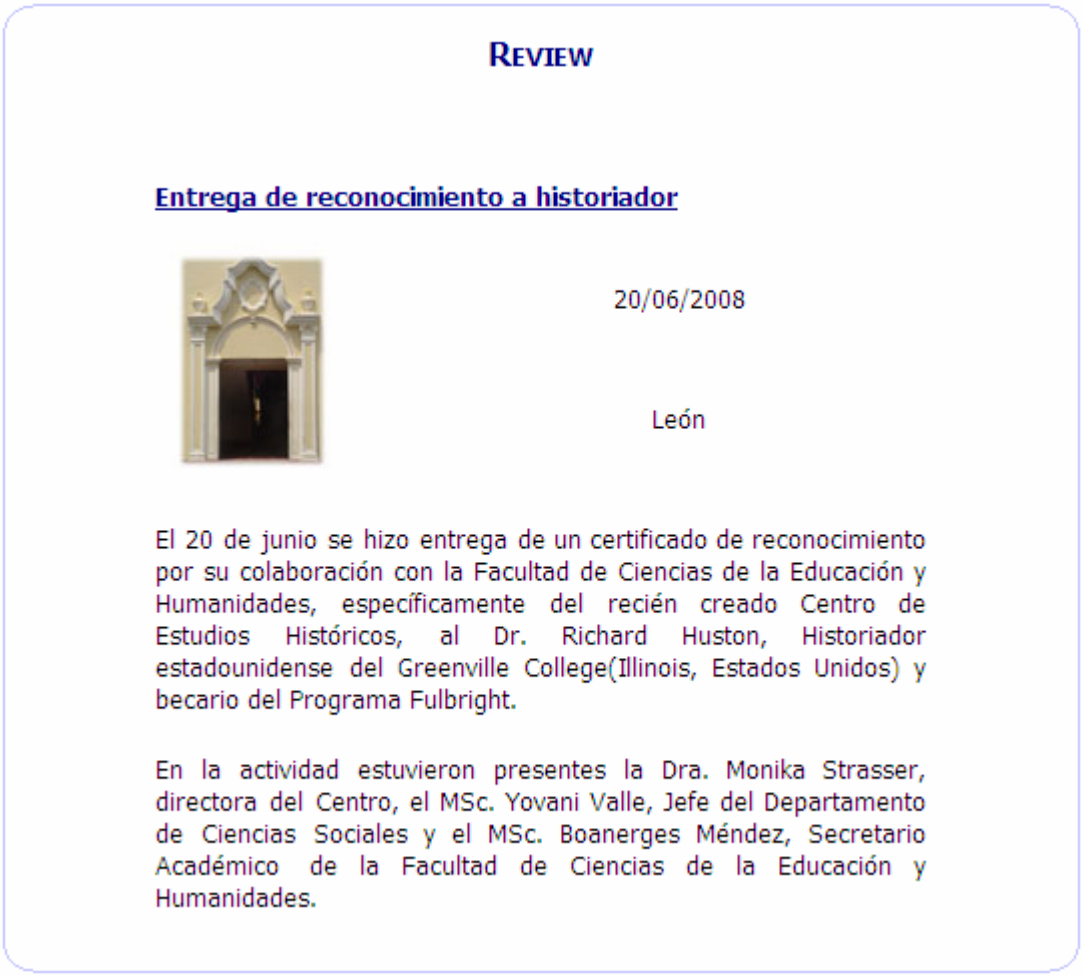


Fig. 80 Página Review.

6.10. Página Encuestas Anteriores

6.10.1. Descripción

Anteriormente se ha visto como en el margen derecho de la página principal se ubicaba un control con la Encuesta más reciente: la encuesta de portada. Pero también puede darse el caso de que existan otras encuestas activas en el portal, no tan recientes. Si se desea votar por ellas, y de paso echarle un vistazo a los resultados de las encuestas cuyo periodo de votación ya ha finalizado, tan sólo se tendrá que hacer clic en el botón “Encuestas anteriores” situado en el control Encuestas común a todas las páginas.

El mecanismo de votación de esta nueva página será idéntico al del control anterior, es decir, se debe estar previamente registrado, y tan sólo se podrá emitir un voto por encuesta.

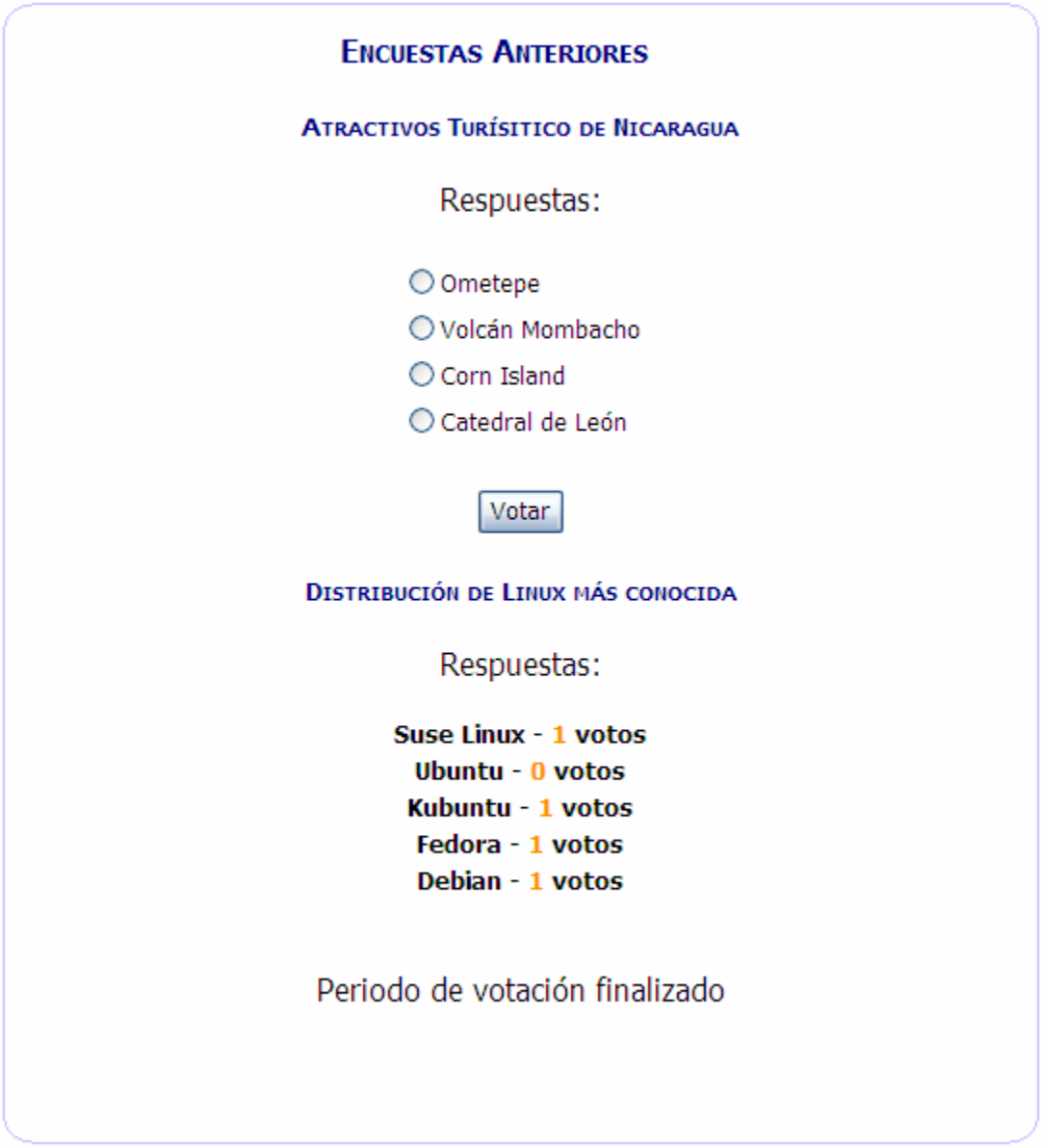


Fig. 81 Página Encuestas Anteriores.

6.10.2. Desarrollo

Esta página estará formada por un control denominado EncuestasAnteriores.ascx (ubicado en la carpeta Encuesta), el cual se describirá en primer lugar para posteriormente centrarse en la propia página.

6.10.2.1. Control EncuestasAnteriores.ascx

Está formado por un control Repeater(rp_Encuestas) cuyo origen de datos (DataSourceID), es un objeto ObjectDataSource (ODSEncuesta), cuyo método asociado a la operación Select es SelectList del objeto comercial BLLEncuesta. El código asociado al control Repeater es el siguiente:

```
<asp:Repeater ID="rp_Encuestas" runat="server" DataSourceID="ODSEncuesta"
OnItemDataBound="rp_Encuestas_ItemDataBound"
OnItemCommand="rp_Encuestas_ItemCommand" >
<HeaderTemplate></HeaderTemplate>
    <ItemTemplate>
     <asp:UpdatePanel ID="UpdatePanel1" runat="server" EnableViewState="true">
        <ContentTemplate>
            <asp:Label ID="etPregunta" runat="server" CssClass="titulospequeños"
Text='<%# Eval("_Pregunta")%>'
                ForeColor="DarkGreen" SkinID="titulosmaster"></asp:Label>
            <br />
            <br />
            <asp:Label ID="etantesrespuestas" runat="server" CssClass="etiquetas"
Text="Respuestas:"></asp:Label>
            <br />
            <asp:Label ID="etrespuestas" runat="server" CssClass="etiquetaspequeñas"
Text=" "
                Font-Bold="true" Visible="false"></asp:Label>
            <asp:Label ID="etdespuesrespuestas" runat="server" CssClass="etiquetas"
Text="<br /><br />Periodo de votación finalizado"
                Visible="false"></asp:Label>
            <br />
            <div align="center">
                <asp:RadioButtonList ID="rb_Respuestas" runat="server" CssClass="textos">
                </asp:RadioButtonList></div>
            <br />
            <asp:Button ID="btnVotar" runat="server" Text="Votar" CssClass="textoscenter"
Visible="false"  CommandName="votar"/>
            <asp:HiddenField ID="hfIdEncuesta" runat="server" />
            <br />
            <asp:Label ID="lb_error" runat="server" ForeColor="Red" Visible="false"
CssClass="textos"></asp:Label>
            <asp:Label ID="lb_soloreg" Text="<br/> Sólo puedes votar si estas registrado
<br/>"
                CssClass="textos" runat="server" ForeColor="Red"
Visible="false"></asp:Label>
            <asp:Label ID="lb_msg" CssClass="textos" runat="server" ForeColor="Red"
Visible="false"></asp:Label>
            <br />
        </ContentTemplate>
     </asp:UpdatePanel>

</ItemTemplate>
</asp:Repeater>
```

Como se puede observar, el Repeater estará formando por una serie de elementos ItemTemplate que se ubicarán en el interior de un UpdatePanel para evitar así la recarga de toda la página. Cada uno de dichos elementos estará formado por la pregunta de la encuesta que se mostrará en forma de etiqueta, así como por otra etiqueta de la que subyacerán las distintas respuestas en forma de RadioButtonList.

A continuación se dispone de un botón con el que se podrá insertar el voto, y por último un par de etiquetas que se mostrarán sólo en caso de error. Para recoger todos los datos a mostrar y realizar las operaciones pertinentes, se recurren a los siguientes métodos:

```
protected void rp_Encuestas_ItemDataBound(object sender, RepeaterItemEventArgs e)
 {
    Encuesta  miencuesta;
    string strRes = "<br />";
    if (e.Item.DataItem != null)
    {
       miencuesta = (Encuesta)e.Item.DataItem;
       Label etrespuestas = (Label)e.Item.FindControl("etrespuestas");
       Label etdespuesrespuestas = (Label)e.Item.FindControl("etdespuesrespuestas");
       if (miencuesta._Fecha_Cierre < DateTime.Now)
       {
          ListRespuestaEncuesta  respuestas = new ListRespuestaEncuesta();
          respuestas =
BLLRespuestaEncuesta.ListByIdEncuesta(miencuesta._Id_Encuesta);
          foreach (RespuestaEncuesta respuesta in respuestas)
          {
             int votos =
BLLRespuestaEncuestaUsr.GetNumVotosByIdRespuesta(respuesta._Id_Respuesta);
             strRes += respuesta._Texto+ " - <Font color='darkorange'>" + votos.ToString()
+ "</font> votos" + "<br />";
          }
          e.Item.FindControl("rb_Respuestas").Visible = false;
          e.Item.FindControl("btnVotar").Visible = false;
          etrespuestas.Text = strRes;
          etrespuestas.Visible = true;
          etdespuesrespuestas.Visible = true;
       }
       else
       {
          ListRespuestaEncuesta respuestas = new ListRespuestaEncuesta();
          respuestas =
BLLRespuestaEncuesta.ListByIdEncuesta(miencuesta._Id_Encuesta);

          RadioButtonList rbl = ((RadioButtonList)e.Item.FindControl("rb_Respuestas"));
          rbl.DataSource = respuestas;
          rbl.DataTextField = "_Texto";
          rbl.DataValueField = "_Id_Respuesta";
          rbl.DataBind();
          ((HiddenField)e.Item.FindControl("hfIdEncuesta")).Value =
miencuesta._Id_Encuesta.ToString();
```

```
                rbl.Visible = true;
                Button btnvotar = (Button)e.Item.FindControl("btnVotar");
                btnvotar.Visible = true;
                if (HttpContext.Current.User.Identity.Name.ToString() == "")
                {
                    btnvotar.Enabled = false;
                    ((Label)e.Item.FindControl("lb_soloreg")).Visible = true;
                }
            }
        }
    }
```

Primeramente se obtienen las respuestas a las encuestas y se generan los enlaces correspondientes para rellenar los radiobuttons y poder así votar.

Si la fecha de la encuesta en cuestión es anterior a la del día de hoy, o la encuesta está inactiva, significará que se deben deberemos mostrar las respuestas acompañadas del número de votos que han recibido, y por tanto descartar la posibilidad de que se pueda votar por ellas. Empleando pues los métodos ListByIdEncuesta para hallar cada una de las respuestas y GetNumVotosByIdRespuesta para averiguar su número de votos asociado. Se mostrarán con la ayuda de un bucle y se ocultarán los Radiobuttons y el botón votar para dejar únicamente visibles las etiquetas de las respuestas. Tras aplicarles ciertos atributos, su apariencia resultará como las imágenes expuestas anteriormente.

En caso contrario, es decir, la encuesta aún sigue activa, se obtienen las repuestas a través del identificador de encuesta y dicha colección de elementos es asociada con la fuente de datos de los elementos que se mostrarán. Se rellena de esta forma todas las propiedades de la colección de elementos RadioButton. Se hace también visible el botón Votar, aunque su disponibilidad dependerá de si quien intenta votar esta registrado o no. En caso negativo, se hará visible la etiqueta que indica que sólo se puede votar si se está registrado.

En cuanto al propio proceso de votación se realiza de la siguiente manera:

```
protected void rp_Encuestas_ItemCommand(object source, RepeaterCommandEventArgs e)
    {
        //Vamos a votar sobre la encuesta
        RespuestaEncuestaUsr respuesta = new RespuestaEncuestaUsr();
        respuesta._Id_Encuesta =
int.Parse(((HiddenField)e.Item.FindControl("hfIdEncuesta")).Value);
        int idrespuesta =
int.Parse(((RadioButtonList)e.Item.FindControl("rb_Respuestas")).SelectedValue);
```

```
        string usuario = HttpContext.Current.User.Identity.Name;
        bool havotado =
BLLRespuestaEncuestaUsr.UsrHaVotado_IdEncuesta(respuesta._Id_Encuesta, usuario);

        if (havotado)
        {
            Label lb_error = (Label)e.Item.FindControl("lb_error");
            lb_msg.Text = "";
            lb_msg.Visible = false;
            lb_error.Text = "Ya has votado en esta encuesta con anterioridad";
            lb_error.Visible = true;
        }
        else
        {
            Label lb_msg = (Label)e.Item.FindControl("lb_msg");
            respuesta._Id_Respuesta = idrespuesta;
            respuesta._Usuario= usuario;
            BLLRespuestaEncuestaUsr.AddUpdate(respuesta);
            lb_msg.Text = "Tu voto ha sido registrado";
            lb_msg.Visible = true;
        }
    }
```

Se crea un nuevo objeto de tipo RespuestaEncuestaUsr y se almacena en él el identificador de la respuesta. Se localiza el control y también el RadioButton marcado. A continuación se verifica si ese usuario había votado previamente. De ser así, se le indica mediante el mensaje de error correspondiente. Se deberá hacer visible dicho control. En caso contrario, es decir, si la operación se ha realizado con éxito, e muestra el mensaje: **Tu voto ha sido registrado**, y se procede a guardar el voto, llamando para ello al método AddUpdate de la clase BLLRespuestaEncuestaUsr.

6.10.2.2. Página EncuestasAnteriores.aspx

Finalmente, cree una nueva página que denominada EncuestasAnteriores.aspx y en la que se incluirá, un Panel y dentro de este el control anteriormente descrito.

6.11. Página Enlaces

6.11.1. Descripción

Dando clic en la pestaña Enlaces del menú principal, se muestra una página cuyo objetivo es mostrar los Enlaces de interés y que están, a los que se podrá acceder con tan sólo dar clic sobre uno de ellos.

6.11.2. Desarrollo

Esta sección está compuesta principalmente de un control de usuario (Enlaces.ascx) y de la página propiamente dicha (Enlaces,aspx), tanto el control como la página deberá añadirlos en una carpeta denominada Enlaces, que cuelga de la carpeta Controles del sitio Web.

6.11.2.1. Control Enlaces.ascx

Está compuesto de un control Repeater (Repeater1) cuyo origen de datos es un objeto de tipo ObjectDataSource (ObjectDataSource1), cuya operación Select está asociada al método SelectListActivos

El código del control Repeater es el siguiente:

```
<asp:Repeater ID="Repeater1" runat="server" DataSourceID="ObjectDataSource1"
EnableViewState="False" >
<HeaderTemplate>
        <table>
    </HeaderTemplate>
    <ItemTemplate>
      <tr>
        <td style="100%">
            <asp:Image ID="Image1" runat="server"
ImageUrl='<%#("~/Imagenes/shortcut.gif")%>' />
            <a target="_blank"  title="<%# Eval("_Url") %>" href="<%# Eval("_Url")%>">
             <%# Eval("_TextoUrl")%></a>
             <br />
             <asp:Label ID="etDescripción" runat="server" Text='<%#
Eval("_Descripción")%>' Cssclass="textos"  Width="100%">
             </asp:Label>
```

```
            <br />
             <br />
          </td>
        </tr>
    </ItemTemplate>
    <FooterTemplate>
      </table>
    </FooterTemplate>
</asp:Repeater>
```

Está formado principalmente por una imagen a modo de viñetas, descargue dicha imagen desde el Portal oficial de la asignatura y realice los mismos pasos que con la imagen de noticias RSS, acompañada de la dirección URL del enlace, y de una etiqueta relacionada con el campo _Descripción.

6.11.2.2. Página Enlaces.aspx

Cree una nueva página Web (Enlaces.aspx), añada un control Panel1 al que se le aplicarán las esquinas redondeadas, dentro de este agregue otro Panal (Panel2), que albergará un etiqueta con el texto “Enlaces de Interés”, (CssClass=etiquetas, SkinID = titulosmaster), finalmente agregue dentro del Panel1, el control Enlaces.ascx creado anteriormente.

6.12. Página Mostrar Resultados

6.12.1. Descripción

Página destina a mostrar el resultado de la búsqueda que el usuario, haya realizado en el control buscador.

6.12.2. Desarrollo

Esta sección está compuesta de una única página, que deberá incluir en la subcarpeta Google de la carpeta Controles.

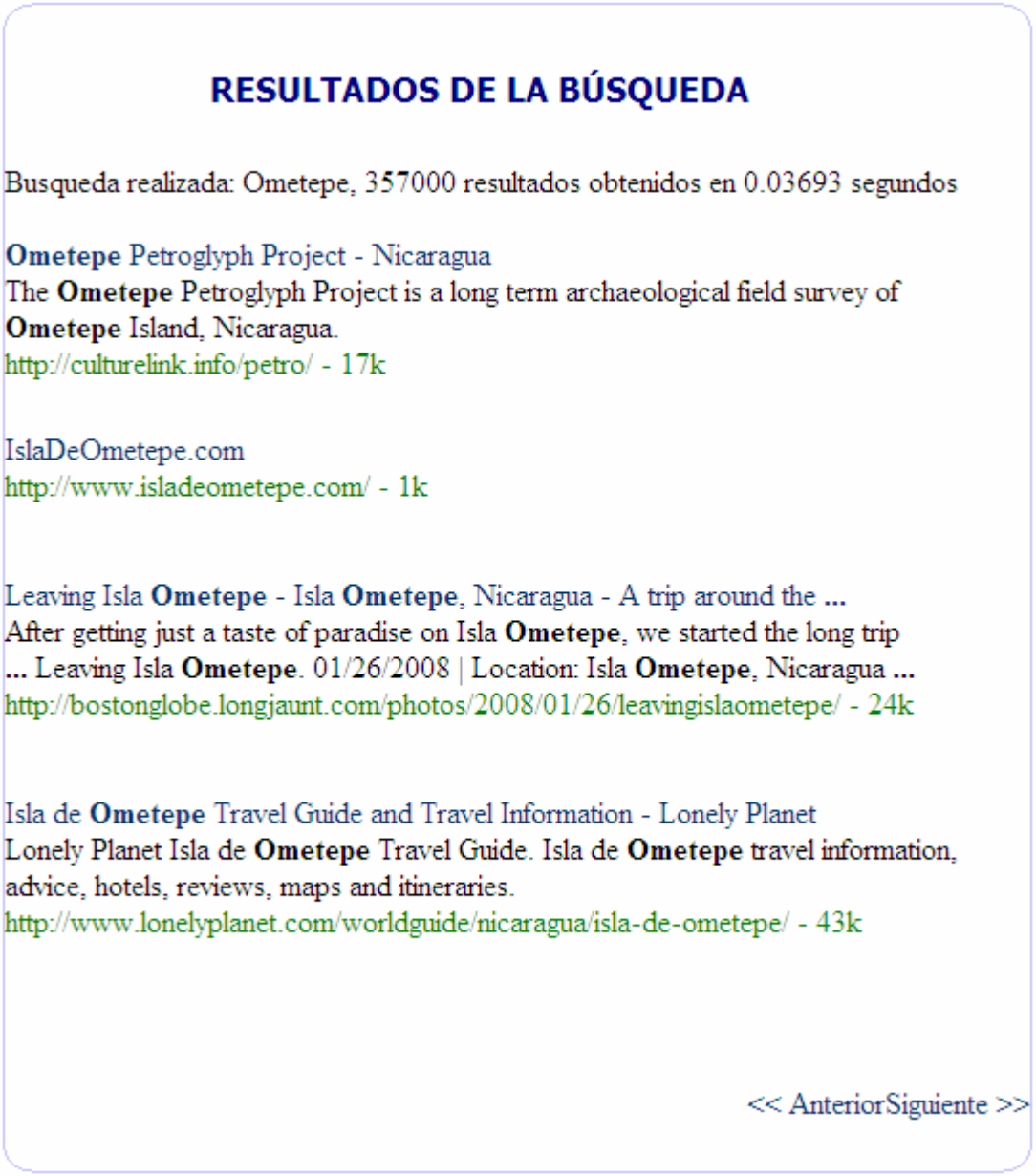


Fig. 82 Resultados de la búsqueda.

6.12.2.1. Página MostrarResultados.aspx

Añada un Panel al cual se le aplicarán las esquinas redondeadas, en el interior de este panel coloque una etiqueta con el texto “RESULTADOS DE LA BÚSQUEDA”, esta dentro de otro panel, a continuación añada un control Xml (del espacio de nombres System.Web.UI.WebControls), y establezca su propiedad TransformSource al archivo ~/Controles/Google/google.xsl, cuyo código se muestra a continuación.

```
<xsl:stylesheet version = '1.0'
   xmlns:xsl='http://www.w3.org/1999/XSL/Transform'>
  <xsl:template match="resultElements">
    <!-- transform main results -->
    <xsl:for-each select="ResultElement">
      <a href="{URL}" target="_blank">
        <xsl:value-of select="./title" disable-output-escaping="yes"/>
      </a>
      <br />
      <xsl:if test="./snippet!="">
        <xsl:value-of select="./snippet" disable-output-escaping="yes"/>
        <br/>
      </xsl:if>
      <div class="letraepequena" style="color:#008000"  >
        <xsl:value-of select="./URL" disable-output-escaping="yes"/>
        -   <xsl:value-of select="./cachedSize"/>
      </div>
      <br />
      <br />
    </xsl:for-each>
  </xsl:template>
  <xsl:template match="GoogleSearchResult">
    <!-- search summary -->
    Busqueda realizada: <xsl:value-of select="searchQuery" />, <xsl:value-of
select="./estimatedTotalResultsCount"/>
    resultados obtenidos en     <xsl:value-of select="./searchTime"/> segundos
    <br />
    <br />
    <xsl:apply-templates select="resultElements"/>
  </xsl:template>
</xsl:stylesheet>
```

Finalmente añada un par de hipervínculos, los cuales le permitirán al usuario navegar hacia atrás o hacia delante en los resultados de la búsqueda. El diseño deberá ser similar al mostrado en la siguiente figura:

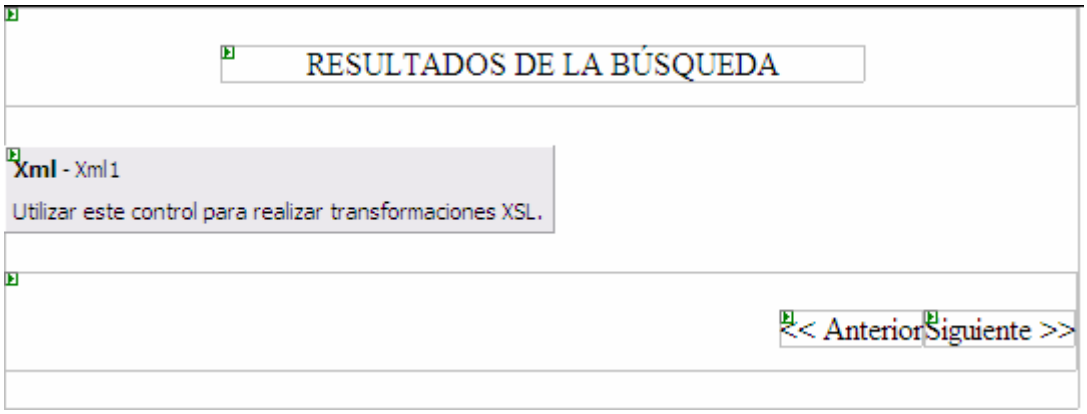


Fig. 83 Diseño de la página MostrarResultados.aspx

Por último nos falta editar el evento Load de la página, el cual se encargará de llamar al servicio de Google y mostrará los resultados al usuario, para ello debe primero añadir una referencia Web al servicio de Google (*Ver Anexos, Construyendo un motor de búsqueda para el sitio Web utilizando el Web Service de Google*)

```
public partial class Controles_Google_MostrarResultados : BasePage
{
   private const string GOOGLEKEY = "wgLzmMFQFHKyFcdsnuJ2QTXv+qKoEWcY";
   protected void Page_Load(object sender, EventArgs e)
   {
      string cadenabuscar = Request["cadena"];
      bool global = (Request["ambito"] == "1");
      int indice = 0;
      if (!IsPostBack)
      {
         //hago la llamada a google
         GoogleSearch.GoogleSearchService s = new GoogleSearch.GoogleSearchService();
         try
         {
            if (Request["indice"] != null)
            {
               indice = int.Parse(Request["indice"]);
            };
            GoogleSearch.GoogleSearchResult r = s.doGoogleSearch(GOOGLEKEY,
               cadenabuscar, indice, 10,
               false, "", false, "", "", "");
            System.Xml.Serialization.XmlSerializer xs =
                  new System.Xml.Serialization.XmlSerializer(r.GetType());
            System.IO.StringWriter sw = new System.IO.StringWriter();
            xs.Serialize(sw, r);
            //lo transformamos con el fichero XSLT
            Xml1.DocumentContent = sw.ToString();
            hlkSiguiente.Visible = true;
            if (indice == 0)
            {
               hlkAnterior.Visible = false;
               hlkSiguiente.NavigateUrl = "?ambito=" + Request["ambito"] + "&cadena=" +
                  cadenabuscar + "&indice=" + (indice + 10).ToString();
            }
            else
            {
               hlkAnterior.Visible = true;
               hlkAnterior.NavigateUrl = "?ambito=" + Request["ambito"] + "&cadena=" +
                  cadenabuscar + "&indice=" + (indice - 10).ToString();
               hlkSiguiente.NavigateUrl = "?ambito=" + Request["ambito"] + "&cadena=" +
                  cadenabuscar + "&indice=" + (indice + 10).ToString();
            }
         }
         catch    { }
      }
   }
}
```

6.13. Página Foros

6.13.1. Descripción

Sin duda una de las cosas más importantes del sitio Web es el Foro. En él los usuarios podrán compartir opiniones, iniciar discusiones, debatir sobre algún tema, etc. En definitiva, representa el mejor escenario de encuentro y participación. Para poder participar en el foro es indispensable haberse registrado previamente, si bien esto no será necesario si tan sólo se pretende echar una ojeada a los mensajes existentes.

Fig. 84 Página Foros.

En la página principal del foro (Clic sobre la pestaña Foros del menú principal), se mostrarán los foros actualmente existentes junto a una pequeña descripción, y acompañados por el número de temas y mensajes que los componen.

Si se hace clic en alguno de los foros, se abrirá una nueva página con un listado de los temas existentes en éste en el que se puede apreciar el número de respuestas o posts que tiene cada uno, así como la fecha de creación y el autor del mismo, y también los datos relativos a la última actividad del tema.

Fig. 85 Página Mensajes.

Si se desea iniciar un nuevo tema, bastará con hacer clic en el botón de abajo para que se despliegue una nueva página en la que se puede escribir el texto que desee.

En ella se puede seleccionar el tipo de tema (general, de información o de alerta), lo que hará que junto a cada uno de ellos se sitúe un pequeño icono tal y como muestra la imagen expuesta a continuación.

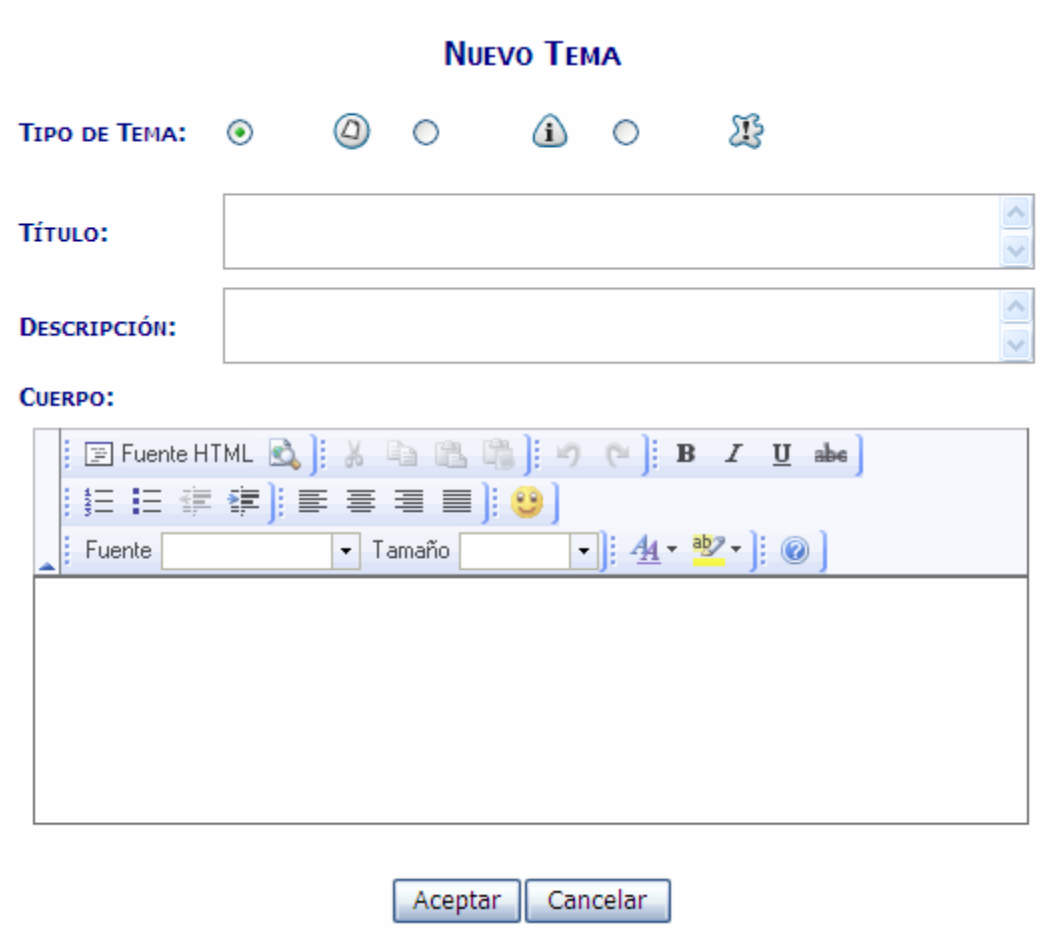


Fig. 86 Añadir un tema en un Foro.

Además, para el cuerpo del mensaje se cuenta con la ayuda del Fckeditor (*Ver Anexos: Editor de Texto: FCKeditor*), que además de posibilitar la edición de texto, en esta ocasión permitirá también hacer uso de emoticones.

Por otro lado, si se hace clic en cualquiera de los temas, se puede ver un listado con todos los mensajes añadidos.

En la parte superior se verá cuando se emitió el mensaje y el tema o asunto al que pertenece.

En el margen izquierdo se mostrarán datos referentes a usuario que envió el mensaje, como su nombre, el número total de mensajes emitidos, la fecha en la que se registró o su estado actual.

Por último, si lo que desea es añadir un nuevo mensaje, debe hacer clic en el botón habilitado para ello, y acto seguido se desplegará una página muy similar a la de la creación de nuevo tema, en la que podrá agregar el comentario. Cabe mencionar que en caso de que el tema aparezca con un icono al lado en forma de candado, significará que dicho tema se encuentra cerrado, y que por tanto no se puede añadir nuevos mensajes.

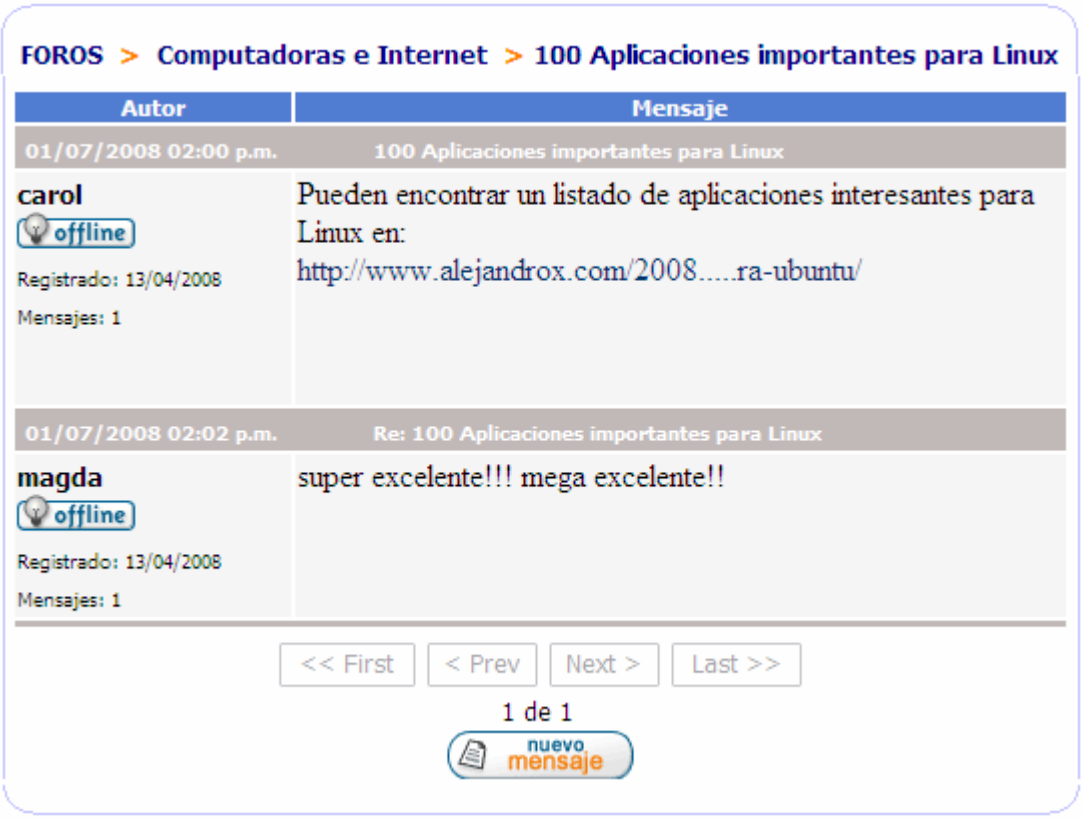


Fig. 87 Página Mensajes de un Foro.

6.13.2. Modo Moderador

Para facilitar las labores de gestión y administración del foro, se ha diseñado un nuevo perfil. Se trata del perfil moderador (esto será tratado más adelante). Cuando un usuario con dichas facultades accede al foro, observará una serie de símbolos junto a cada tema o foro, que le permitirán, haciendo clic sobre ellos, cerrar un determinado foro o tema, o volver a dejarlo activo y abierto. No podrá sin embargo eliminarlos de forma definitiva, pues eso es tarea que atañe al propio administrador del portal.

Fig. 88 Foros en modo Moderador.

Asimismo, podrá crear nuevos foros desde la página general de foros. También podrá ocultar aquellos mensajes dentro del foro que considere poco apropiados dentro de la temática a tratar, para ejercer así cierto control sobre los contenidos que se mostrarán. Para ello tan sólo deberá hacer clic en el botón en forma de aspa situado en el margen izquierdo de cada mensaje. En ese caso, el post será sustituido por un mensaje como el siguiente:

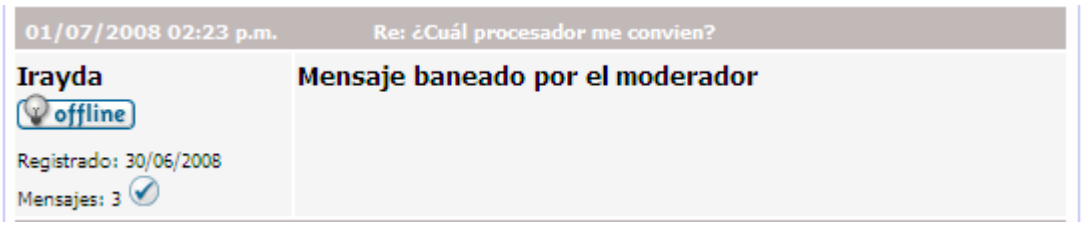


Fig. 89 Banear un mensaje.

6.13.3. Desarrollo

Para la creación de la interfaz de usuario del foro se hace uso de cuatro páginas: IndiceForos.aspx, Foro.aspx, Mensajes.aspx y Añadir.aspx. Los tres primeros se añadirán en una carpeta Foro que cuelga de la carpeta Controles y para la última se debe crear una carpeta Participar dentro de la carpeta Foro.

6.13.3.1. Página IndiceForos.aspx

Para la creación de esta página, como en la mayor parte de las páginas del sitio Web, en primer lugar añada un panel en cuyo interior se coloca la mayoría de los elementos de la misma junto con un extender RoundedCorners, que aportara un diseño de esquinas redondeadas al mismo. En esta clase utilizará los siguientes espacios de nombres:

```
using System.Collections.Generic;
using MembershipUtilities;
```

A continuación coloque una Label cuyo texto será el título de la página (Foros Actuales, CssClass= titulosgrandes, SkinID = titulosmaster). Seguidamente añada control de tipo UpdatePanel y dentro de este un ObjectDataSource y configúrelo de tal forma que el método asociado a la consulta SELECT sea Foro_ListPaginados de la clase ForoBLL y como método para el recuento total de filas utilice Foro_Num.

El código es el siguiente:

```
<asp:ObjectDataSource id="ObjectDataSource1" runat="server"
MaximumRowsParameterName="pageSize" StartRowIndexParameterName="pageIndex"
EnablePaging="True" TypeName="PortalWeb.BLL.BLLForo" DeleteMethod="Delete"
SelectMethod="Foro_ListPaginados" OldValuesParameterFormatString="original_{0}"
DataObjectTypeName="PortalWeb.BO.Foro" UpdateMethod="AddUpdate"
SelectCountMethod="Foro_Num">
<SelectParameters></SelectParameters>
</asp:ObjectDataSource>
```

Para mostrar los resultados de la consulta utilice un GridView, en el cual se establece el como campo clave el campo _Id_Foro y además active la paginación.

A continuación edite sus columnas. Interesará tener una columna para la imagen asociada al foro, otra para su nombre, una cuarta tercera para mostrar el número de mensajes principales o temas que pertenecen a dicho foro, una cuarta para mostrar el número total de mensajes registradas en el foro, otra columna con la información referente al último mensaje almacenado y una columna final para que el moderador pueda activar o desactivar el foro. Todas estas columnas serán del tipo TemplateField. Además se dejarán las columnas _Id_Foro, _Fecha_Creación y _Creador, todas de tipo BoundField y su propiedad Visible a False.

	Foro	Temas	Mensajes	Último mensaje	
	DataBound DataBound	DataBound	DataBound	DataBound	
	DataBound DataBound	DataBound	DataBound	DataBound	
	DataBound DataBound	DataBound	DataBound	DataBound	

La primera columna simplemente constará de un control Image cuya propiedad ImageUrl irá vinculada al campo _Activo. Si está activo se verá la imagen True.gif almacenada en la carpeta ~/Controles/Foro/Imagenes, por el contrario si está desactivado se verá la imagen False.gif. Recuerde descargar estas imágenes y guardarlas en el lugar adecuado. (ImageUrl = Eval("_Activo", "~//Controles//Foro//Imagenes//{0}.gif")).

En la segunda columna, sitúe una tabla con dos filas y una columna. En su primera fila coloque un HyperLink cuya propiedad Text irá asociada con el campo _Nombre y su propiedad NavigateUrl estará enlazada al campo _Id_Foro de la siguiente manera:
Eval("_Id_Foro", "~//Controles//Foro//Foro.aspx?id={0}").Constituyendo así un enlace para la pagina especifica de el foro en cuestión. En la segunda fila sitúe una etiqueta que estará destinada a mostrar la descripción del foro, con lo que su propiedad Text estará enlazada al campo _Descripción.

Las dos siguientes columnas estarán constituidas por una etiqueta cada una de ellas. La primera tendrá su propiedad Text enlazada con la propiedad _Id_Foro que servirá de parámetro al método GetNumTemas, con lo que el texto a mostrar será el resultado de la llamada a este método. La segunda, llamará al método GetNumMens (GetNumMens(Eval("_Id_Foro")).

El código correspondiente a cada uno de estos métodos es el siguiente:

```
protected string GetNumTemas(object idforo)
   {
      return BLLMensaje.NumPrincipalesById_Foro((int) idforo).ToString();
   }

protected string GetNumMens(object idforo)
   {
      return BLLMensaje.NumById_Foro((int)idforo).ToString();
   }
```

La quinta columna también está formada únicamente por una etiqueta, en este caso, el campo _Id_Foro se utilizará como parámetro de entrada del método GetUltimaActividad. Éste devuelve el resultado de llamar al método UltimaFechaById_Foro de la clase BLLMensaje.

La última columna solo la verán aquellos usuarios que tengan el rol de moderador y constará de dos ImageButton, que debe configurar de la siguiente manera:

```
<asp:TemplateField ShowHeader="False"><ItemTemplate>
<asp:ImageButton id="ImageButton1" runat="server"
ImageUrl="~/Controles/Foro/Imagenes/icon6.gif" Visible='<%# GetActivo1(Eval("_Activo"))
%>' __designer:wfdid="w7" CommandArgument='<%# Eval("_Id_Foro") %>'
CommandName="Borrar" OnClientClick="return confirm ('¿Estas seguro de querer borrar
este foro?');" OnCommand="cerrar_foro" CausesValidation="False"></asp:ImageButton>
<asp:ImageButton id="ImageButton2" runat="server"
ImageUrl="~/Controles/Foro/Imagenes/icon5.gif" Visible='<%# GetActivo2(Eval("_Activo"))
%>' __designer:wfdid="w8" CommandArgument='<%# Eval("_Id_Foro") %>'
CommandName="Reactivar" OnClientClick="return confirm ('¿Estas seguro de querer
reactivar este foro?');" OnCommand="activar_foro"
CausesValidation="False"></asp:ImageButton>
</ItemTemplate>
</asp:TemplateField>
```

En primer lugar se configura la propiedad Visible de cada uno de los botones, la cual estará enlazada al campo Activo. En el primer botón se depende del resultado de llamar a la función GetActivo1 y en el segundo de llamar a la función GetActivo2. De esta manera el botón situado a la izquierda se verá cuando el foro esté activo, y el de la derecha cuando esté desactivado. En ambos casos se necesita que el método GetVisible esté a true. Dicho método devuelve true si el usuario que ha iniciado sesión pertenece al rol de moderador. A continuación se muestra el código de los tres métodos mencionados.

```
protected bool GetVisible()
{
   string user = HttpContext.Current.User.Identity.Name.ToString();
   if (user == "")
```

```
        return false;
    else
        return (HttpContext.Current.User.IsInRole("Moderadores"));
    }

protected bool GetActivo1(object activo)
    {
        bool visible = GetVisible();
        bool respuesta = visible & (bool)activo;
        return respuesta;
    }

protected bool GetActivo2(object activo)
    {
        bool visible = GetVisible();
        bool respuesta = visible & !(bool)activo;
        return respuesta;
    }
```

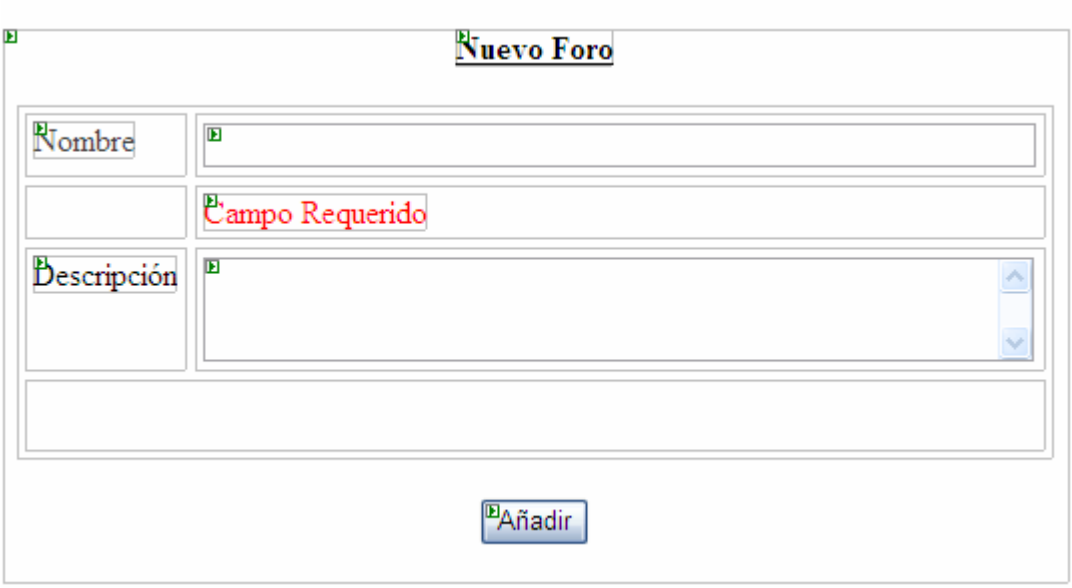
A continuación, edite los manejadores del evento OnCommand de ambos botones, teniendo como argumento asociado al comando el campo _Id_Foro.

```
protected void cerrar_foro(object sender, CommandEventArgs e)
    {
        Foro miforo;
        int id = Convert.ToInt32(e.CommandArgument.ToString());
        miforo = BLLForo.SelectGetById(id);
        miforo._Activo = false;
        BLLForo.AddUpdate(miforo);
        ListMensaje listamensajes = new ListMensaje();
        listamensajes = BLLMensaje.ListHuerfanosById_Foro(miforo._Id_Foro);
        foreach (Mensaje mimensaje in listamensajes)
        {
            mimensaje._Activo = false;
            BLLMensaje.AddUpdate(mimensaje);
        }
    }

protected void activar_foro(object sender, CommandEventArgs e)
    {
        Foro miforo;
        int id = Convert.ToInt32(e.CommandArgument.ToString());
        miforo = BLLForo.SelectGetById(id);
        miforo._Activo = true;
        BLLForo.AddUpdate(miforo);
    }
```

En el primer método se pone la propiedad _Activo del foro en cuestión a False, lo que conlleva además la desactivación de todos los temas que cuelgan del mismo. En el segundo método

simplemente se activa el foro en cuestión. Si se desea volver a activar los temas que cuelgan de él deberá hacerlo tema por tema desde la página Foro.aspx.

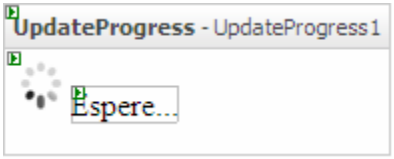
Finalmente, y para posibilitar la adición de nuevos foros añada dentro otro panel (Panel2) en cuyo interior colocará una etiqueta con el texto “Nuevo Foro” y debajo una tabla con tres filas y dos columnas. Este panel sólo será visible por el moderador, para ello en el método Page_Load añada la siguiente sentencia: Panel2.Visible = GetVisible();



Como se ve en la imagen, existe una tabla con dos cajas de texto donde se puede escribir el nombre del nuevo foro y su descripción. Pulsando el botón situado en la parte inferior del panel se añadirá el foro. Recuerde definir la propiedad ValidationGroup del botón y del campo de validación, que estará asociado a la caja de texto correspondiente al nombre del foro. El manejador del evento Clic del botón, añade un foro de la siguiente manera:

```
protected void añadir_foro(object sender, EventArgs e)
    {
        Foro miforo = new Foro();
        miforo._Nombre = tb_nombreforo.Text;
        miforo._Descripción = tb_descrforo.Text;
        miforo._Fecha_Creación = DateTime.Today;
        miforo._Creador = HttpContext.Current.User.Identity.Name.ToString();
        miforo._Activo = true;
        tb_nombreforo.Text = "";
        tb_descrforo.Text = "";
        BLLForo.AddUpdate(miforo);
        GridView1.DataBind();
    }
```

Todos los controles (GridView, ObjectDataSource y Panel2), están dentro del UpatePanel, al que deberá asociar un control UpdateProgress.



Añada también antes del panel destinado a añadir un nuevo Foro por parte del moderador una etiqueta (Label6), destinada a mostrar el número de mensajes que están actualmente en línea, añada para ello en el manejador del evento Page_Load de la siguiente sentencia:

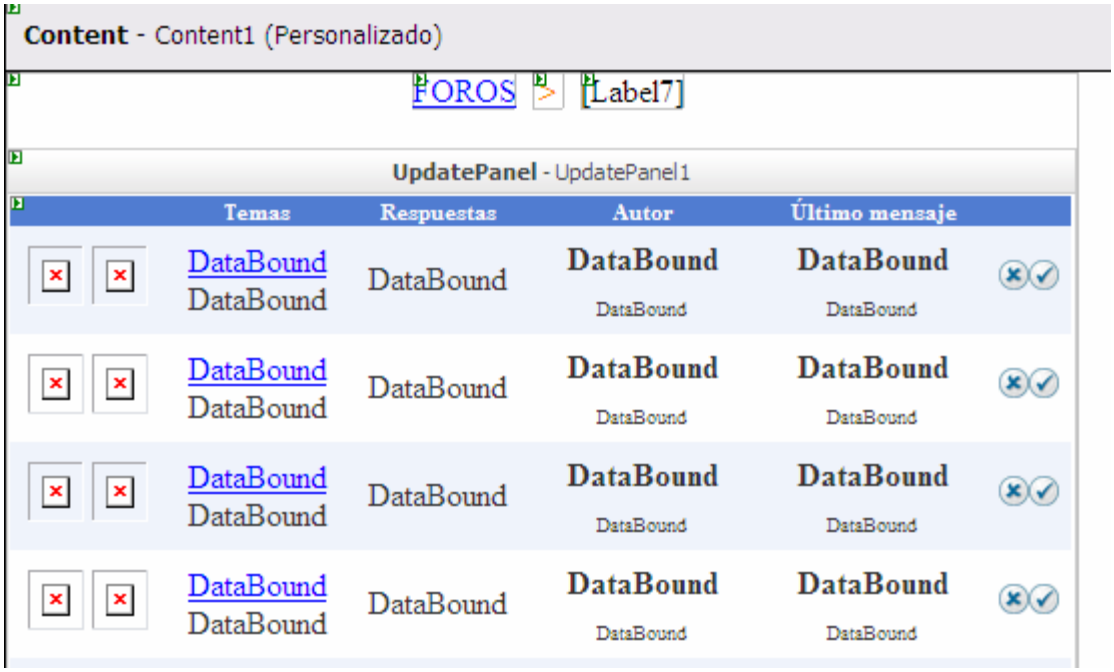
```
protected void Page_Load(object sender, EventArgs e)
   {
     Label6.Text = "Número de usuarios registrados online: " +
Membership.GetNumberOfUsersOnline();
}
```

6.13.3.2. Página Foro.aspx

En esta página se visualizarán los temas (mensajes principales) de un foro determinado. Para ello, en primer lugar se agregará un ObjectDataSource, asociando el método ListHuerfanosById_ForoPaginados como método para la operación Select. El parámetro de entrada de dicho método corresponde con el id del foro en cuestión y vendrá determinado por una QueryString.

Seguidamente agregue un GridView para mostrar el listado de temas así como algunas propiedades más de cada uno de ellos. Será bastante similar al de la página IndiceForos.aspx, por lo que sólo se destacarán algunos aspectos del mismo

En este caso tendrá seis columnas, la primera para la imagen asociada al tema, la segunda para el título y descripción del tema, la tercera indicará el número de respuestas asociadas al mismo, la cuarta mostrará información sobre el autor y la hora de creación, la quinta hará lo mismo con la última respuesta y la sexta exclusivamente será visible por el moderador y le servirá al mismo para activar o desactivar un tema.



Esta vez la primera columna estará formada por dos imágenes, siendo visible únicamente una de ellas.

```
<asp:TemplateField ShowHeader="False"><ItemTemplate>
<asp:Image id="Image1" runat="server" ImageUrl='<%# Eval("_Tipo_Mensaje",
"~//Controles//Foro//Imagenes//{0}.gif") %>' __designer:wfdid="w6" Visible='<%#
GetActivo1(Eval("_Activo")) %>'></asp:Image>
<asp:Image id="Image2" runat="server" ImageUrl='<%# Eval("_Activo",
"~//Controles//Foro//Imagenes//{0}.gif") %>' __designer:wfdid="w7" Visible='<%#
!GetActivo1(Eval("_Activo")) %>'></asp:Image>
</ItemTemplate></asp:TemplateField>
```

El código la propiedad Visible de ambas nunca será igual. Una imagen se verá cuando el tema este activo y corresponderá con el tipo de mensaje. Y la otra cuando este desactivados y corresponderá con la imagen de tema cerrado. La función GetActivo1 es la misma que la de la página IndiceForos.aspx. Por otro lado, la manera de activar y desactivar temas por parte del moderador es similar a la página IndiceForos.aspx.

Una vez configurado el GridView añada un ImageButton en la parte inferior de la página para poder añadir nuevos temas al foro. Al evento clic de este botón le asociaremos un manejador.

```
protected void añadir_tema(object sender, ImageClickEventArgs e)
    {
        Response.Redirect("~//Controles//Foro//Participar//Añadir.aspx?idforo=" +
```

```
idforo.ToString());
    }
```

Este manejador nos enviará a la página Añadir.aspx pasando como parámetro QueryString el identificador del foro.

Para facilitar al usuario la navegación por el foro añadiremos en la parte superior de la página un hipervínculo que nos redirija a la página principal del foro (IndiceForos.aspx) y una etiqueta que nos indique el nombre del foro del cual estamos visualizando los temas.


```
int idforo;
    string nombre;
    protected void Page_Load(object sender, EventArgs e)
    {
        idforo = Convert.ToInt32(Request.QueryString["id"].ToString());

        if (!IsPostBack)
        {
            Foro miforo = new Foro();
            miforo = BLLForo.SelectGetById(idforo);
            nombre = miforo._Nombre;
            ImageButton1.Visible = miforo._Activo;
        }

        Label7.Text = nombre;
        Page.Title = "Foro: " + nombre;
}
```

En el código, si el foro está cerrado el botón que nos posibilita la inserción de nuevo temas está oculto. Al igual que en la página anterior colocaremos todos los elementos en el interior de un UpdatePanel que tendrá asociado a su vez un UpdateProgress. Recuerde configurar el resto de las columnas, llamando en cada caso al método correspondiente de la clase BLLMensaje.

6.13.3.3. Página Mensaje.aspx

En esta página se visualizan todas las respuestas asociadas a un determinado mensaje, mostrando dicho mensaje en primer lugar. Para ello se harás uso de un control Repeater. En él veremos una lista de mensajes, así como información del usuario que introdujo cada uno de ellos.

En la parte superior de la pantalla, al igual que en la página anterior agregue dos hipervínculos, uno que nos lleve a la página principal del foro y otro al tema en el que está el mensaje cuyas respuestas estamos visualizando. A parte tendremos una etiqueta que nos mostrará el nombre del mensaje.

Para mostrar los mensajes dentro del Repeater, editaremos sus plantillas de la siguiente manera:

```
<asp:Repeater id="Repeater1" runat="server"  EnableViewState="False" >
    <HeaderTemplate>
          <table style="width: 529px;">
            <thead>
                 <tr align="left" style="background-color:#507CD1">
                      <td align="center" valign="top">
                         <asp:Label ID="Label2" runat="server" CssClass="textosnegrita"
Text="Autor" ForeColor="White" Font-Size="11px"></asp:Label></td>
                        <td style="width: 383px" align="center" valign="top">
                          <asp:Label ID="Label3" runat="server" CssClass="textosnegrita"
Text="Mensaje" ForeColor="White" Font-Size="11px"></asp:Label></td>
                    </tr>
               </thead>
               </HeaderTemplate>
                    <ItemTemplate>
                      <tr>
                        <td align="left" valign="top"  colspan="2" style="font-weight:bold;
background-color:#C1B8B8"> 
                            <asp:Label ID="FechaLabel" runat="server" CssClass="textospeq"
Text='<%# Eval("_Fecha","{0:g}") %>' ForeColor="White"></asp:Label>
                            
                          <asp:Label ID="TituloLabel" runat="server" Text='<%#
Eval("_Título") %>' CssClass="textospeq" ForeColor="White"></asp:Label>
                        </td>
                    </tr>
                <thead>
                  <tr  style="background-color:#F5F5F5">
                    <td align="left" valign="top">
                      <asp:Label ID="UsuarioLabel" runat="server" Text='<%#
Eval("_Usuario") %>' CssClass="textosnegrita"></asp:Label>
                  <br />
                    <asp:Image ID="Image1" runat="server"
ImageUrl='<%#GetImagenEstadoUser( Eval("_Usuario")) %>' />
                  <br />
                  <asp:Label ID="MailLabel" runat="server" Text='<%#GetCuandoRegistrado(
Eval("_Usuario")) %>' CssClass="textospeq"> </asp:Label>
                  <br />
                  <asp:Label ID="NumMenLabel" runat="server"
Text='<%#NumMensajesUser( Eval("_Usuario")) %>' CssClass="textospeq"></asp:Label>
                  <asp:ImageButton id="IButton1" runat="server"
```

```
Visible='<%#GetVisible1(Eval("_Visible")) %>'
ImageUrl="~/Controles/Foro/Imagenes/icon6.gif" CausesValidation="False"
CommandArgument='<%# Eval("_Id_Mensaje") %>' CommandName="Borrar"
OnClientClick="return confirm ('¿Estas seguro de querer ocultar este mensaje?');"
OnCommand="cerrar_mensaje"></asp:ImageButton>
                    <asp:ImageButton id="IButton2" runat="server"
Visible='<%#GetVisible2(Eval("_Visible"))%>'
ImageUrl="~/Controles/Foro/Imagenes/icon5.gif" CausesValidation="False"
CommandArgument='<%# Eval("_Id_Mensaje") %>' CommandName="Reactivar"
OnClientClick="return confirm ('¿Estas seguro de quieres hacer visible este mensaje?');"
OnCommand="activar_mensaje"></asp:ImageButton>
          <td style="width: 383px" align="left" valign="top">
                  <asp:Label ID="TextoLabel" runat="server" Text='<%# Eval("_Texto") %>'
Visible='<%#GetVisible(Eval("_Visible")) %>' ></asp:Label>
                    <asp:Label ID="ModeradorLabel" runat="server" Text="Mensaje baneado
por el moderador" CssClass="textosnegrita" Visible='<%#!GetVisible(Eval("_Visible"))
%>'></asp:Label>
                    <br /> <br /> </td>
            </tr>
          </thead>
        </ItemTemplate>
        <FooterTemplate>
      </table>
     </FooterTemplate>
</asp:Repeater>
```

Los datos se mostrarán haciendo uso de la paginación, por lo que tendrá que crear las propiedad pageIndex, pageSize y TotalRowCount (este retornará: return BLLMensaje.NumById_Padre(idmensaje) + 1) así como la interfaz de paginación formada por los cuatro botones. Además añada una etiqueta cuyo ID será “CurrentPageNumber”, destinada a informar al usuario de qué página está visualizando.

Por otro lado, cree dos métodos, uno llamado Cargar_Datos() y otro llamado Select() cuyo código será el siguiente:

```
private void CargarDatos()
   {
     mensajes  =  BLLMensaje.ListById_PadrePaginados(idmensaje, pageIndex, pageSize);
       Select();
       this.Repeater1.DataSource = mensajes;
       this.Repeater1.DataBind();
   }

private void Select()
   {
       FirstPage.Enabled = pageIndex != 0;
       PrevPage.Enabled = pageIndex != 0;
```

```
        int LastPageStartRowIndex = ((TotalRowCount - 1) / pageSize) * pageSize;
        NextPage.Enabled = pageIndex < LastPageStartRowIndex;
        LastPage.Enabled = pageIndex < LastPageStartRowIndex;
        int pagtot;

        if (TotalRowCount < pageSize || TotalRowCount == pageSize)
            pagtot = 1;
        else if ((TotalRowCount % pageSize) == 0)
            pagtot = (TotalRowCount / pageSize);
        else
            pagtot = (TotalRowCount / pageSize) + 1;
        CurrentPageNumber.Text = numpag.ToString() + " de " + Convert.ToString(pagtot);
    }
```

Cada vez que se haga clic en uno de los botones de la interfaz el manejador de este evento llamará al método Cargar_Datos(). Además también llame al mismo en el método Page_Load.

```
public partial class Controles_Foro_Mensaje : BasePage
{
    int idforo;
    int idmensaje;
    string nombrepadre;
    string nombreforo;
    int numpag = 1;
    ListMensaje mensajes;

    protected void Page_Load(object sender, EventArgs e)
    {

        idmensaje = Convert.ToInt32(Request.QueryString["idmensaje"].ToString());
        idforo = BLLMensaje.GetId_ForoById_Mensaje(idmensaje);

            Mensaje padre = new Mensaje();
            padre = BLLMensaje.SelectGetById(idmensaje);
            Foro miforo = new Foro();
            miforo = BLLForo.SelectGetById(idforo);
            nombreforo = miforo._Nombre;
            nombrepadre = padre._Título;
            ImageButton1.Visible= padre._Activo;

        HyperLink1.Text = nombreforo;
        HyperLink1.NavigateUrl = "~//Controles//Foro//Foro.aspx?id=" + idforo.ToString();
        Page.Title=Label3.Text = nombrepadre;
        CargarDatos();
    }
```

En cuanto a la información reflejada por el Repeater, se podrá saber si el creador de un mensaje está en línea en offline mediante el método GetImagenEstadoUser.

```
protected string GetImagenEstadoUser(object nombre)
    {
        MembershipUser usuario = Membership.GetUser((string)nombre);
        if (usuario.IsOnline)
            return "~//Controles//Foro//Imagenes//online.gif";
        else
            return "~//Controles//Foro//Imagenes//offline.gif";
    }
```

También se mostrará su mail, cuando se registró en el sitio y cuántos mensajes ha escrito.

```
protected string GetMailUser(object nombre)
    {
        MembershipUser usuario = Membership.GetUser((string)nombre);
        return usuario.Email;
    }

    protected string GetCuandoRegistrado(object nombre)
    {
        MembershipUser usuario = Membership.GetUser((string)nombre);
        return "<bold>Registrado: </bold> " + usuario.CreationDate.ToShortDateString();
    }

    protected string NumMensajesUser(object nombre)
    {
         return "<bold>Mensajes: </bold> " +
BLLMensaje.NumByUsuario((string)nombre).ToString();
    }
```

En lo que se refiere al contenido del mensaje, el moderador podrá decidir si es apto o no para que aparezca en el foro. Para se agregó un control ImageButton con los métodos OnCommand adecuados, define estos de manera similar que en la página Foros.aspx, pero al final llame de cada uno llame al método CargarDatos().

Para determinar si este botón es visible o no, define la siguiente función, y llámela en el resto de métodos GetVisible que necesite el repeater.

```
protected bool GetVisible()
    {
        string user = HttpContext.Current.User.Identity.Name.ToString();
        if (user == "")
            return false;
        else return (HttpContext.Current.User.IsInRole("Moderadores"));
    }
```

Si el moderador pulsa sobre el mismo, cambiará la propiedad Visible del mensaje en cuestión a false, provocando que en lugar de visualizarse el contenido del mismo se visualice un mensaje

que indica que hay sido “borrado”. Para que esto suceda así tendremos dos etiquetas, la primera enlazada al campo Texto y la segunda con dicho mensaje. Si nos fijamos en la siguiente figura, sus propiedades Visible nunca serán igual. Por ejemplo:

```
protected bool GetVisible1(object visible)
   {
     return ( GetVisible() & (bool)visible);
   }
```

Por último si se desea añadir una nueva respuesta al tema, podremos hacerlo pulsando sobre un ImageButton cuyo manejador del evento clic será el siguiente:

```
protected void añadir_respuesta(object sender, ImageClickEventArgs e)
   {
      Response.Redirect("~//Controles//Foro//Participar//Añadir.aspx?idforo=" +
idforo.ToString() + "&reply=" + idmensaje.ToString());
   }
```

Al igual que en las dos páginas anteriores, los elementos de ésta irán en el interior de un UpdatePanel y este dentro de un panel al que se le aplicarán las esquinas redondeadas.

6.13.3.4. Página Añadir.aspx

Para la inserción de un nuevo tema o mensaje en el foro, se recurre a la página Añadir.aspx, se desea que pueda ser accedida únicamente por los usuarios registrados, por tanto añádala dentro de una subcarpeta (Participar) de la carpeta Foro, al final de este apartado se explicará cómo se consigue que no accedan a ella usuarios que no estén registrado.
Esta página está compuesta principalmente por una tabla que contendrá diversos campos, y que estará ubicada en un panel. Su organización corresponde con la mostrada en la **Fig. 90.**

La primera de las filas estará dedicada a recoger el tipo de tema que se desea insertar. A través de un control de tipo RadioButton, acompañado por diferentes imágenes, se podrá escoger cual de ellos se ajusta más a la naturaleza de nuestro mensaje.

A continuación se encuentran varias cajas de texto para insertar el título y la descripción del mensaje. Por último, y ya destinado a almacenar el cuerpo del mensaje, se cuenta con el ya

anteriormente incluido editor de textos FckEditor. Cada una de las filas descritas contará además con su correspondiente etiqueta para describir cada campo en cada ocasión. Asimismo, y al final de la página, situaremos un par de botones: aceptar y cancelar.

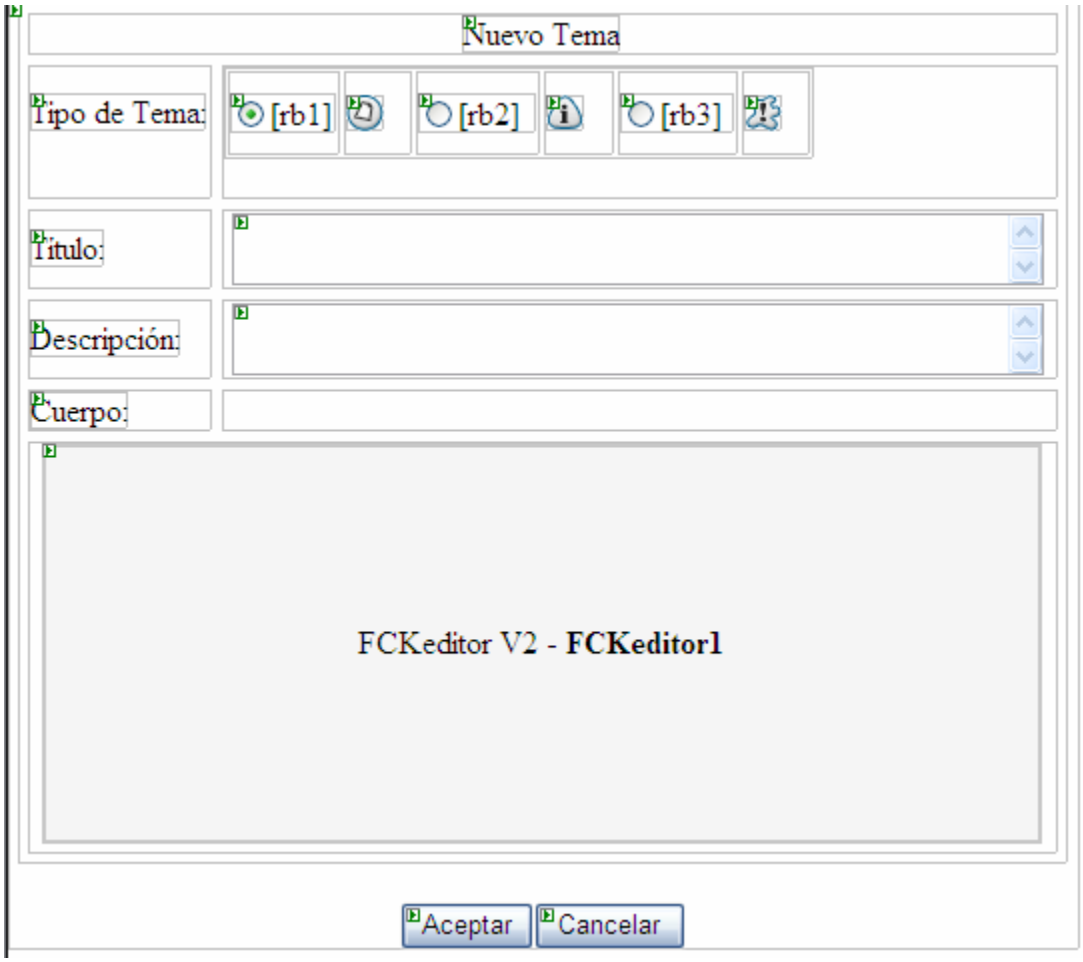


Fig. 90 Añadir un nuevo Tema.

El primero de ellos tendrá asociado como manejador del evento clic el siguiente método:

```
protected void insertar_mensaje(object sender, EventArgs e)
{
    Mensaje mimensaje = new Mensaje();
    mimensaje._Id_Foro = idforo;
    if (reply)
        mimensaje._Id_Padre = idpadre;
    else
        mimensaje._Descripción = ctDescrip.Text;
    mimensaje._Visible = true;
    mimensaje._Activo = true;
    if (reply)
    {
        Mensaje padre = BLLMensaje.SelectGetById(idpadre);
```

```
            mimensaje._Título = "Re: " + padre._Título;
        }
        else
            mimensaje._Título = ctTitulo.Text;

        mimensaje._Texto = FCKeditor1.Value;
        mimensaje._Usuario = HttpContext.Current.User.Identity.Name;
        mimensaje._Fecha = DateTime.Now;
        if (rb1.Checked)
            mimensaje._Tipo_Mensaje = 1;
        else if (rb2.Checked)
            mimensaje._Tipo_Mensaje = 2;
        else
            mimensaje._Tipo_Mensaje = 3;
        int id = BLLMensaje.AddUpdate(mimensaje);
        if (reply)
        {
            Response.Redirect("~//Controles//Foro//Mensaje.aspx?idmensaje=" +
            mimensaje._Id_Padre.ToString());
        }
        else
        {
            Response.Redirect("~//Controles//Foro//Foro.aspx?id=" +
            mimensaje._Id_Foro.ToString());
        }
    }
```

Como se puede podido observar en el código anterior, en primer lugar se crea un nuevo objeto mensaje, y se le añade el identificador del foro al que pertenece a su propiedad _Id_Foro. Si se trata además de un mensaje de contestación a otro ya existente, se agrega además al campo _Id_Padre el valor de aquel de quien depende. De no ser así, añadiremos directamente su descripción. Asimismo, y para ambos casos, marcaremos como “true” las propiedades _Activo y _Visible del mensaje.

En cuanto al título, si se trata de un mensaje derivado de otro, acudiremos al mensaje padre a través del método BLLMensaje.SelectGetById, pasándole como referencia el identificador del padre. De esta forma obtendremos su campo _Título y le asignaremos el mismo valor al campo _Título del nuevo mensaje. De no ser así, directamente lo tomaremos desde el TextBox correspondiente a dicho campo.

En cuanto al resto de campos, el texto será recogido del FCkEditor en cualquiera de los casos, así como la identidad del usuario que va a emitir el mensaje y el momento en el que lo hace.

Por lo que a la colección de radiobuttons respecta, asignaremos un valor entero a cada uno de ellos para discernir así del tipo de mensaje seleccionado. Podremos de esta forma elegir entre, por ejemplo, un mensaje de información, o un mensaje de alerta.

Una vez hecho todo esto, procederemos a salvar el mensaje. Acto seguido, y dependiendo de si es un mensaje de contestación a otro o no, se nos redirigirá a la página donde se encuentra el mensaje padre, o a la página general del foro en el otro caso.

Con el botón cancelar actuaremos exactamente de la misma forma que en lo anteriormente descrito en cuanto al tema de la redirección se refiere, con la única salvedad de que en este caso no se tendrán en cuenta ninguno de los datos introducidos, tal y como muestra el código del manejador expuesto a continuación.

```
protected void cancelar_click(object sender, EventArgs e)
    {
        if (reply)
        {
            Response.Redirect("~//Controles//Foro//Mensaje.aspx?idmensaje=" +
            idpadre);
        }
        else
            Response.Redirect("~//Controles//Foro//Foro.aspx?id=" + idforo);
    }
```

Por otra parte, y al cargar la página, analizaremos si se trata de un caso en el que queramos añadir un nuevo tema, o si por el contrario vamos a contestar a alguno de los mensajes ya existentes, para lo que deberemos examinar la QueryString. Si por ejemplo, nos encontramos con el segundo de los casos, procederemos a asignar el valor false a cada una de las propiedades Visible de todos los campos que conforman la anterior tabla, salvo el control FCkeditor, que albergará el mensaje de contestación. Esto es debido a que dichos campos no son necesarios en el contexto de un mensaje de contestación. También habremos de modificar el texto de la etiqueta de cabecera para indicar que se trata de una nueva respuesta. En el supuesto de que estemos en el primero de los casos, es decir, un nuevo tema en un foro, no deberemos alterar ninguna de dichas propiedades, ya que todas son relevantes.

```
protected void Page_Load(object sender, EventArgs e)
    {
        if (Request.QueryString.Get("reply") != null)
        {
            etDescripcion.Visible = false;
            ctDescrip.Visible = false;
            ctTitulo.Visible = false;
            etTítulo.Visible = false;
```

```
            etTipoTema.Visible = false;
            rb1.Visible = false;
            rb2.Visible = false;
            rb3.Visible = false;
            Image1.Visible = false;
            Image2.Visible = false;
            Image3.Visible = false;
            Label1.Text = "Nueva respuesta";
            idpadre = Convert.ToInt32(Request.QueryString["reply"]);
            idforo = Convert.ToInt32(Request.QueryString["idforo"]);
            reply = true;
        }
        else
            idforo = Convert.ToInt32(Request.QueryString["idforo"]);
    }
```

Para que esta página sea accesible sólo por usuarios registrados, agregue un fichero web.config en la carpeta Participar, y agregue una regla de acceso en la cual se deniegue el acceso a los usuarios anónimos.

```
<system.web>
   <authorization>
     <deny users="?" />
   </authorization>
 </system.web>
```

6.14. Página RSS

6.14.1. Descripción

Una de las secciones más importantes dentro del portal es sin duda el apartado de Noticias RSS. En esta página se recogerán al minuto, las noticias más importantes de cada canal Rss al que está suscrito nuestro portal.

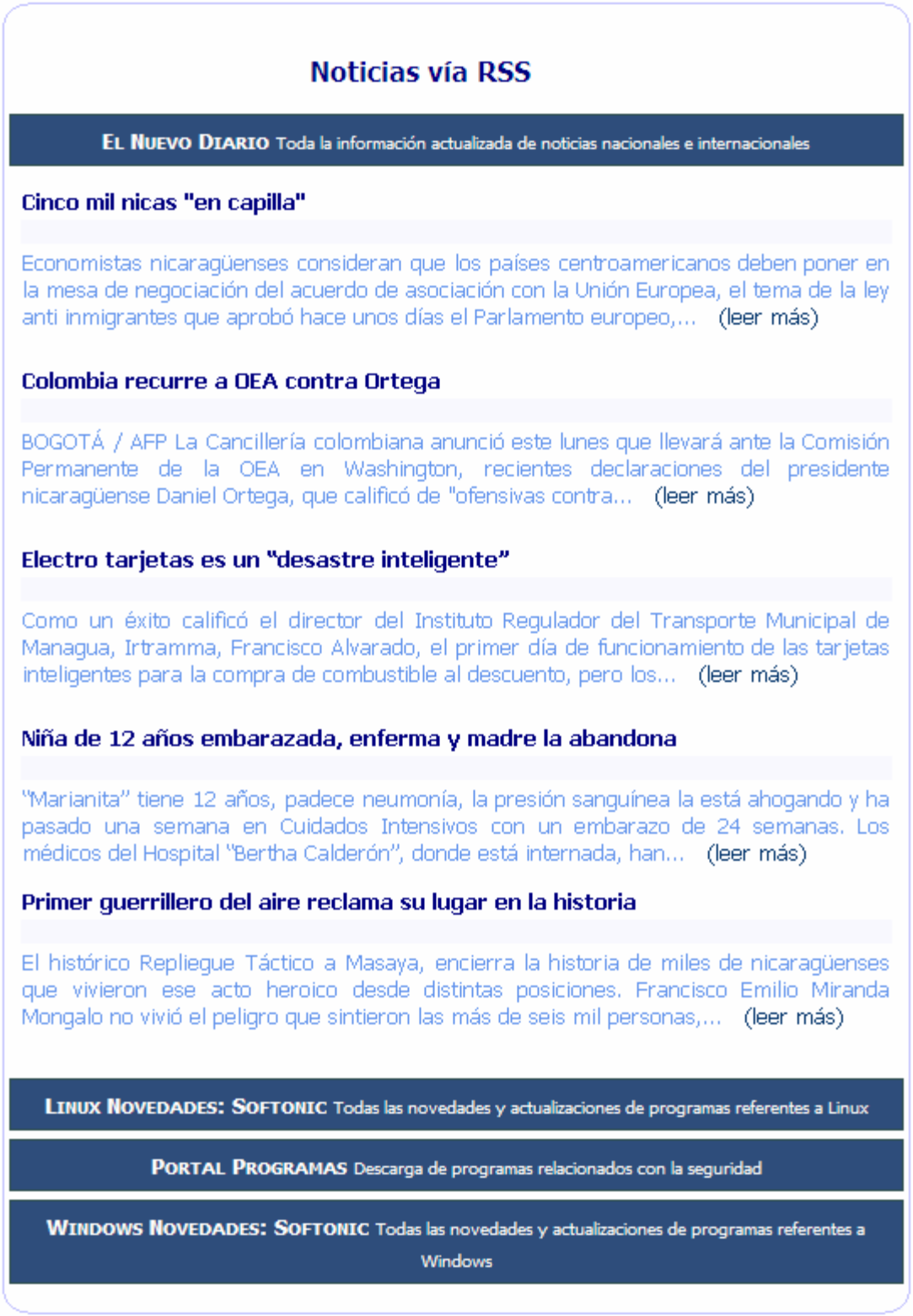


Fig. 91 Página Noticias RSS.

Haciendo clic en cada uno de estos canales, se nos desplegarán todas las noticias asociadas al mismo. Dependiendo de la naturaleza de los elementos que conforman dicha colección de noticias, podremos ver su titular, resumen, o incluso una pequeña imagen. Bastará con pulsar sobre cualquiera de estas noticias para ser remitidos a la auténtica fuente y poder así encontrar toda la información relativa a la misma de forma íntegra.

6.14.2. Desarrollo

Para la realización de esta página cree una carpeta denomina NoticiasRss dentro de la carpeta Controles, dentro de ésta crearemos un control llamado FeedManager.ascx en la carpeta noticiasRSS además de una página que albergará dicho control (NoticiasRss.aspx).

6.14.2.1. Control FeedManager.ascx

El control contará con un objeto ObjectDataSource (odsRSS), que devolverá el listado de FuentesRSS disponibles en cada momento. Para ello, configuraremos como método de la operación SELECT del mismo, el método ObtenerNoticias de la clase BLLCanalRSS.

Para mostrar los datos obtenidos haremos uso de uno de los extender pertenecientes al Control Toolkit de AJAX: el control Accordion. Nos permitirá visualizar cada uno de los elementos de datos proporcionados por la fuente de datos de forma individualizada. De tal forma que, mientras uno de ellos esté siendo mostrado, el resto permanecerán contraídos, pudiéndose tan sólo observar una pequeña cabecera.

A la hora de configurar dicho control nos veremos obligados a hacerlo a través de la vista de código. En primer lugar estableceremos su origen de datos, eligiendo para ello el ObjectDataSource creado. A continuación añadiremos la plantilla HeaderTemplate, donde como podemos apreciar, ubicaremos el título de la fuente de noticias rss.

```
<ajaxToolkit:Accordion ID="Accordion1" runat="server"
ContentCssClass="accordionContent"
            DataSourceID="odsRSS" HeaderCssClass="accordionHeader" Width="530px">
            <HeaderTemplate>
               <asp:Label runat="server" ID="label1" Text='<%#Eval("_title")%>'
CssClass="titulospequeños"
```

```
                    ForeColor="White"></asp:Label>
                    <asp:Label runat="server" ID="label2" Text='<%#Eval("_description")%>'
CssClass="textospeq"
                    ForeColor="White"></asp:Label>
                </HeaderTemplate>
```

Seguidamente añadiremos la plantilla ContentTemplate, en cuyo interior agregaremos un Repeater. El origen de datos de este control será el campo Ítems del objeto de datos FuenteRSS.

```
<ContentTemplate>
    <asp:Repeater ID="childRepeater"
DataSource='<%#DataBinder.Eval(Container.DataItem, "_items")%>'
        runat="server">
```

A continuación personalizaremos la plantilla ItemTemplate. Esta estará formada por una tabla en la que colocaremos los diferentes campos de cada objeto ItemFuenteRSS existente en el campo Ítems.

```
<ItemTemplate>
            <table>
                <tr class="textosnegrita" style="color: Navy" align="left">
                    <td>
                        <%#DataBinder.Eval(Container.DataItem, "_title")%><br />
                    </td>
                </tr>
                <tr style="background-color: #F8F8FF" class="textospeq" align="left">
                    <td>
                        <%#DataBinder.Eval(Container.DataItem, "PubDate")%><br />
                    </td>
                </tr>
                <tr class="textos" style="color: #6495ED">
                    <td>
                        <%#DataBinder.Eval(Container.DataItem, "_description")%>
                          <a target="article" href='<%#
DataBinder.Eval(Container.DataItem, "_link") %>'>
                            (leer más)</a> </br></br>
                    </td>
                </tr>
                </table>
            </ItemTemplate>
        </asp:Repeater>
    </ContentTemplate>
</ajaxToolkit:Accordion>
```

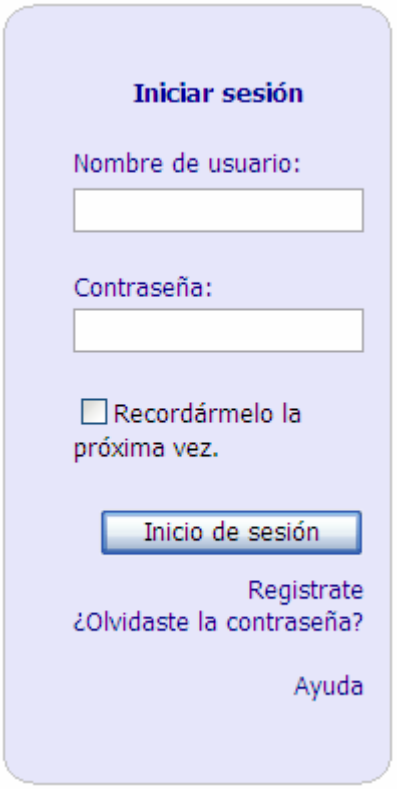
Finalmente añadiremos una tabla con una columna y dos filas. En la primera fila añadiremos una etiqueta para mostrar el titulo de la página, y en la segunda irá situado el control Accordion anterior.

6.14.2.2. Página NoticiasRss.aspx

Una vez creado el control anterior lo ubicaremos en una nueva página denominada NoticiasRSS.aspx, recuerde añadirlo dentro de un panel para aplicarle esquinas redondeadas.

6.15. Controles de sesión ASP .NET

6.15.1. Descripción



Para restringir el acceso a los contenidos, favorecer los mecanismos de participación, y administrar y regular el acceso y funcionamiento del portal, se han incluido una serie de controles de sesión ASP.NET. Todos los controles de esta sección, los deberá incluir en una nueva carpeta llamada Login, que cuelga de la raíz del sitio Web.

El principal de ellos será el control Login situado en la página maestra. Éste cambiará su contenido dependiendo del estado en el que se encuentre el usuario (autenticado o no) y será visible desde todas las páginas del sitio al ser heredado por todas ellas. Además, su aspecto podrá cambiar dependiendo también del tema elegido por cada usuario. Contendrá asimismo un enlace para que los nuevos usuarios puedan registrarse, y otro para que aquellos que olvidaron su contraseña puedan recuperarlas.

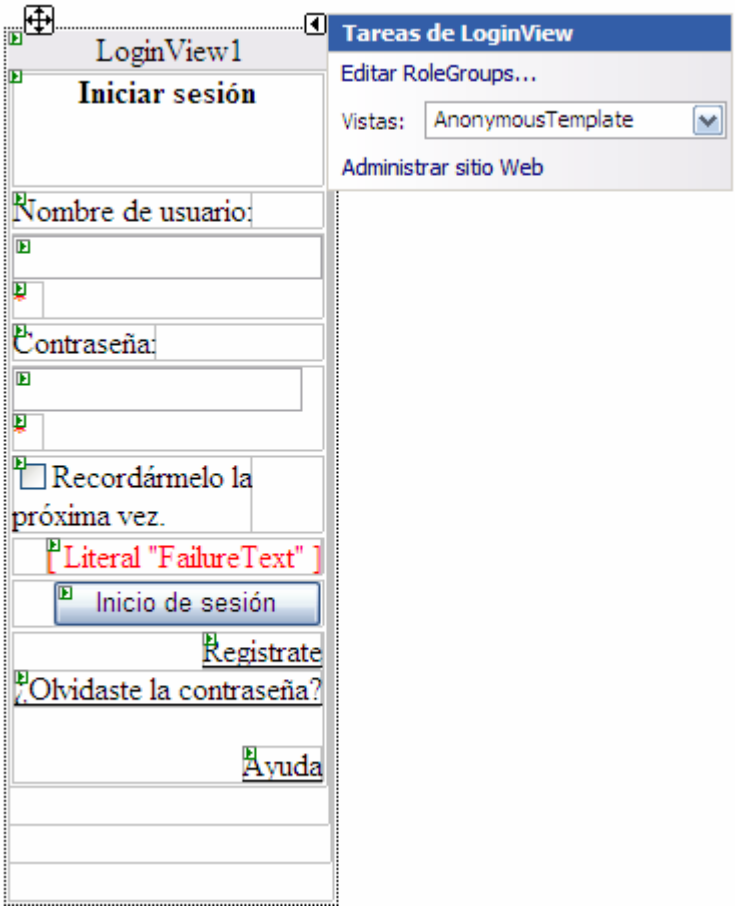
Para la mayoría de contenidos con acceso restringido, cuando intentamos llevar a cabo alguna acción sin habernos autenticado previamente, nos aparecerá un mensaje en el control oportuno que nos indicará que debemos hacerlo. Sin embargo, existen otros casos en los que al intentarlo se nos remite a una página de inicio de sesión cuyo control login se ubicará en la columna central de la página. Es el caso del foro.

6.15.2. Desarrollo

6.15.2.1. Control Login

A diferencia de la mayoría de controles de esta sección, el control Login se encuentra ubicado en una subcarpeta de la carpeta Controles denominada Login.

Es el control más importante de esta sección y más presente en nuestra Web. Para su desarrollo requeriremos del propio control Login y además, de un control LoginView. Éste último nos permitirá personalizar la plantilla dependiendo de si el estado actual de la Web es el de un visitante aún sin autenticar, o un visitante ya autenticado. Recuerde que una vez termine la edición de este control, deberá añadirlo al panel correspondiente en la página maestra.



Para la plantilla anónima del LoginView dispondremos de una estructura como la siguiente:

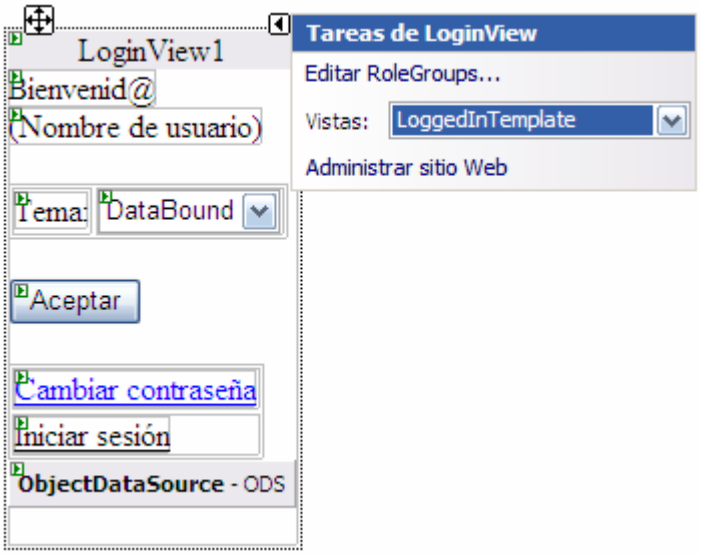
Aprovechando el diseño por defecto del control Login añadiremos algunos campos más para completar su diseño y funcionalidad. Dispondremos de dos cajas de texto para insertar el nombre y la contraseña del usuario, acompañados por dos RequieredFieldValidators que nos indicarán que ambos valores son obligatorios en cualquier caso.

Asimismo contaremos con una casilla que nos permitirá seleccionar si queremos que nuestra contraseña sea recordada cada vez que accedamos al portal, y también con una etiqueta que nos mostrará un mensaje de error en caso de que intentemos acceder con datos erróneos.

Debajo de todos estos controles se sitúan el botón de Inicio de sesión y tres hipervínculos que nos dirigirán a una página de registro para nuevos visitantes (Propiedad NavigateUrl = ~/Login/Registro.aspx), a una página de recuperación de contraseña en caso de que la hayamos olvidado (NavigateUrl = ~/Login/RecoveryPassword.aspx) y a una página de ayuda genérica (NavigateUrl = ~/Login/Ayuda.aspx).

Por otro lado tendremos la plantilla LoggedinTemplate. Esta se compone de diversos elementos. En primer lugar, contamos con una etiqueta en la parte superior con un mensaje de bienvenida. Justo debajo de esta situaremos un control Loginname destinado a mostrar el nombre del usuario actualmente online. Se trata de un control prediseñado que por defecto será capaz de mostrarnos dicho nombre.

A continuación insertaremos una etiqueta cuya propiedad text contendrá la palabra “Tema”, y que irá acompañada de un control DropDownItems destinado a mostrar el conjunto de temas de página disponibles para la elección por parte de los usuarios. Este control contará con el ObjectDataSource correspondiente para proporcionar los datos pertinentes. Más adelante incidiremos en detalle en este aspecto.



Finalmente contaremos con dos hipervínculos. El primero de ellos nos

redirigirá a una página donde podremos cambiar nuestra contraseña. El segundo, nos permitirá cerrar la sesión actual y nos devolverá a la página de inicio, es decir a Default.aspx.

6.15.2.2. Página Registro.aspx

Cuando un usuario hace clic en el enlace “Regístrate” del control Login de la página principal, se le remite a la página de registro. Para su desarrollo emplearemos un control de inicio de sesión de tipo CreateUserWizard (ubicado dentro de un Panel al que se le aplicarán las esquinas redondeadas) cuyas métodos y funcionalidades ya están creadas por defecto, y a las que sólo añadiremos ciertos elementos de personalización.

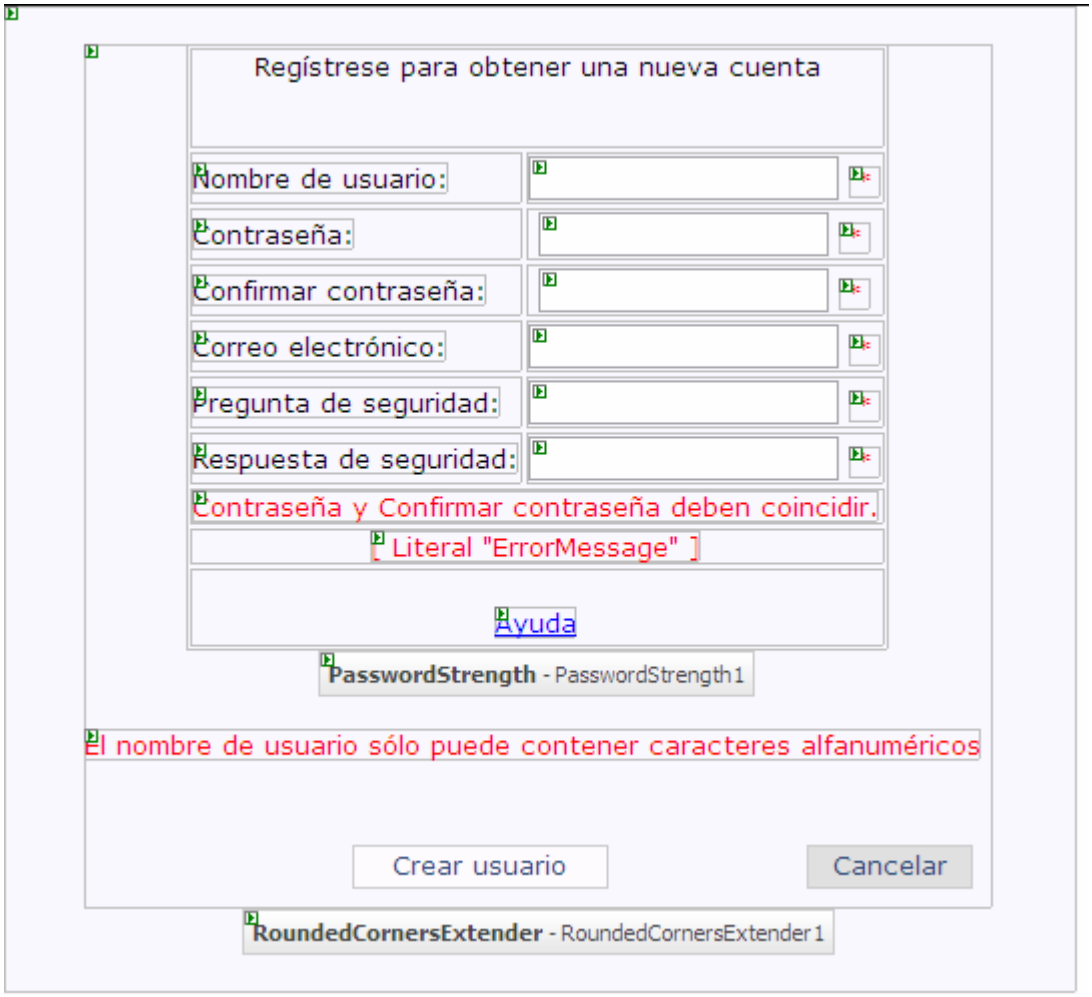
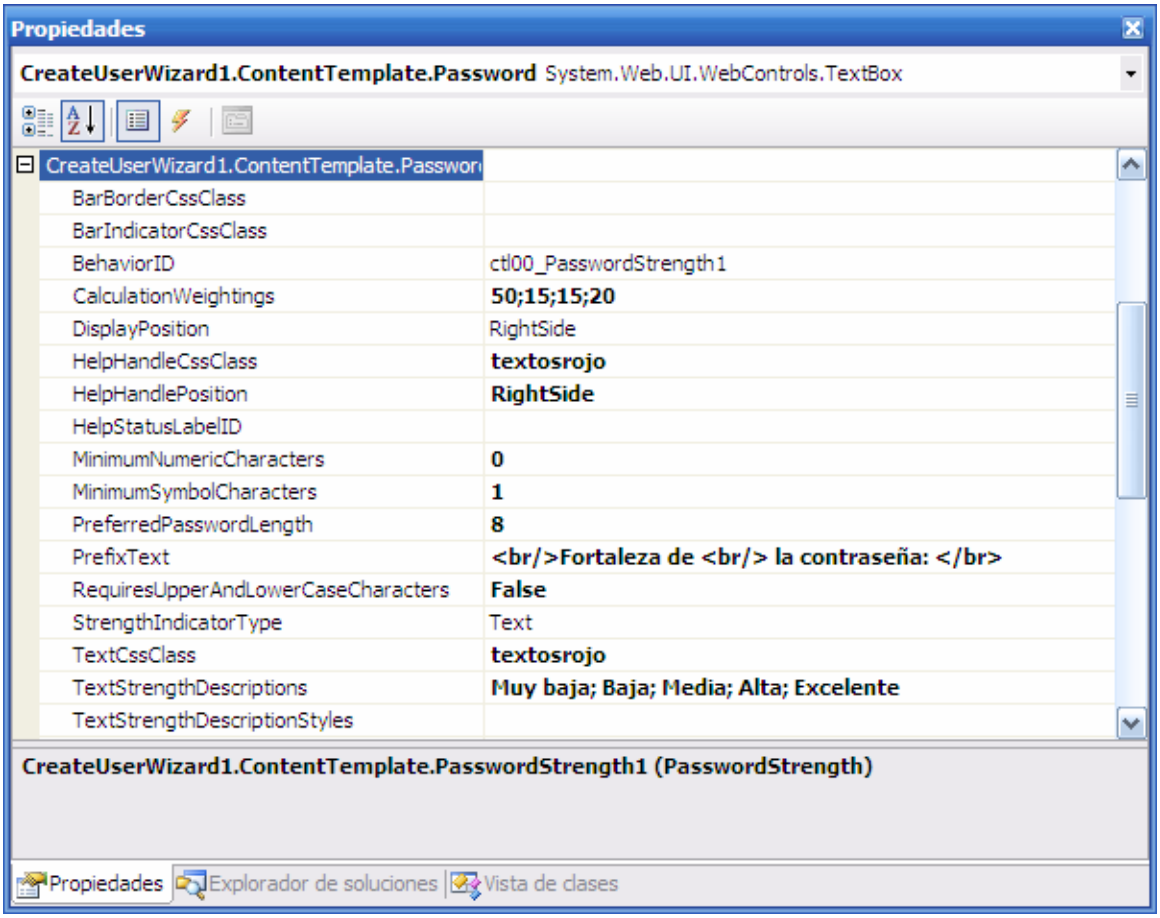


Fig. 92 Diseño página Registro.aspx.

El nuevo usuario deberá introducir de forma obligatoria todos y cada uno de los campos del control, y de no ser así se mostrará un asterisco junto al correspondiente campo a modo de mensaje de error. Asimismo, también deberán de coincidir el campo contraseña con la confirmación de la misma.

Además, se ha añadido un control PasswordStrength procedente del Toolkit de Ajax para mejorar los aspectos de seguridad de la contraseña. Si nos dirigimos a las propiedades de la caja de texto Password y desplegamos la opción Extenders, podremos configurar dichas especificaciones:

Algunos de los valores reseñables de la anterior imagen son, que por ejemplo la longitud mínima de la contraseña será de 8 caracteres, y que al menos uno de ellos deberá ser de tipo alfanumérico. Además conforme vayamos introduciendo la contraseña, un mensaje con letras rojas se nos mostrará en el margen derecho de dicho campo indicándonos cual es la fortaleza de nuestra contraseña, pudiendo variar esta entre muy baja y excelente.



6.15.2.3. Página RecoveryRassword.aspx

En caso de no recordar nuestra contraseña tendremos la posibilidad de recuperarla haciendo clic en la opción correspondiente del menú Login. Se nos redirigirá a una página formada esencialmente por un control de sesión de tipo PasswordRecovery.

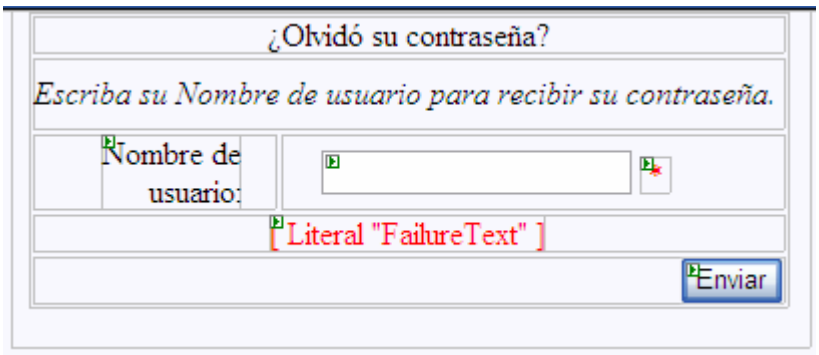
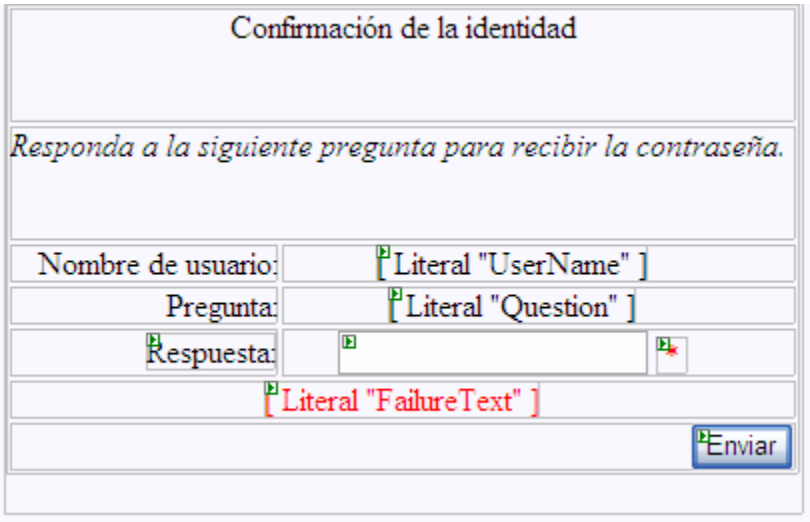


Fig. 93 Recordar la Contraseña.

Éste dispondrá de varias plantillas configurables. La primera de ellas, la de vista de “Nombre de usuario” nos mostrará principalmente una caja de texto en la que insertaremos nuestro nombre de usuario y que obviamente habrá que insertar de forma obligatoria. Una vez pulsemos el botón Enviar, daremos paso a la siguiente vista. (Pregunta)



Por último, y tras haber realizado correctamente los pasos previos, un mensaje nos indicará que se nos ha enviado la nueva contraseña a nuestra cuenta de correo. Cabe mencionar que todos

estos controles se encuentran ubicados sobre paneles a los que se les ha aplicado un estilo de esquinas redondeadas a través de controles RoundedCorners.

6.15.2.4. Página ChangeRassword.aspx

Quizás en alguna ocasión deseemos cambiar nuestra actual contraseña por otra nueva. Recurriremos entonces al enlace “Cambiar contraseña” ubicado en el control Login y disponible sólo en la vista de usuario registrado. Se nos redirigirá entonces a una nueva página cuyo control principal será el control predefinido Changepassword. Lo personalizaremos de tal forma que resulte como en la imagen siguiente, en la vista Cambiar Contraseña.

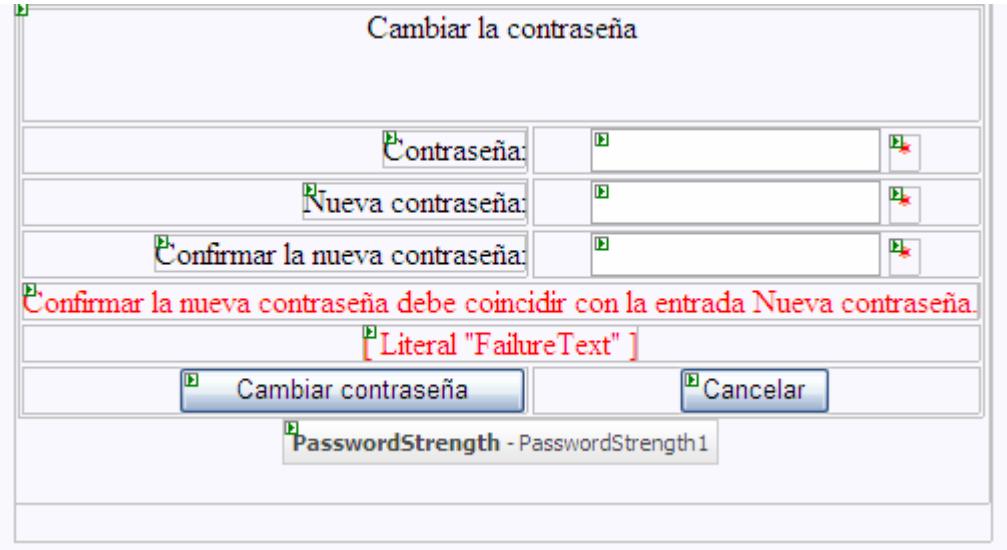
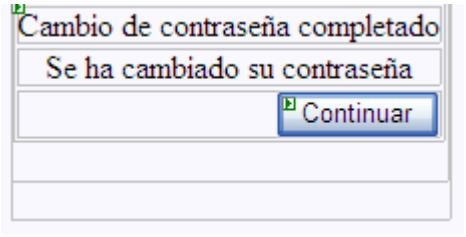


Fig. 94 Cambiar la Contraseña-

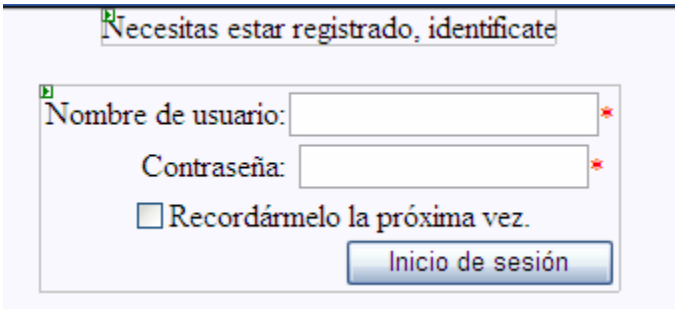
Tan sólo se debe introducir la antigua y la nueva contraseña y hacer clic en “Cambiar contraseña”. En este caso también recurriremos al control PasswordStrength para controlar la fortaleza de la contraseña, y que será configurado de la misma forma que en el caso anterior. El botón cancelar nos llevará a la página Default.aspx.



Una vez realizado el cambio se nos mostrará un mensaje por pantalla que nos indicará que todo se ha realizado correctamente. Haciendo clic en el botón “Continuar” acudiremos de nuevo a la página principal, en la vista Correcto.

6.15.2.5. Página Login.aspx

Por último contaremos también con una página de Login (ubicada en la raíz del sitio Web), que nos surgirá siempre que intentemos acceder a algún contenido restringido que requiera de un registro previo. Es el caso por ejemplo de la página Foro. Si intentamos insertar un nuevo tema o comentario se nos redirigirá inmediatamente a dicha página para que nos identifiquemos.



En ella tan sólo aparecerá un control de tipo Login similar al empleado en la página maestra (pero más sencillo), y en el que, tras introducir correctamente nuestros datos, nos remitirá al contenido de acceso restringido del que proveníamos.

6.16. Regulación y acceso a contenidos

6.16.1. Introducción

Como en todo sitio Web, es muy importante regular y controlar el acceso a los diferentes contenidos y elementos que conforman nuestra aplicación. Para ello hemos apostado por una regulación basada en roles, y cuyas líneas generales aparecen gráficamente descritas en la siguiente imagen. En ella podemos observar qué contenidos están al alcance de cada quién, y cuales son los perfiles o roles existentes, cuya definición será detallada a continuación.

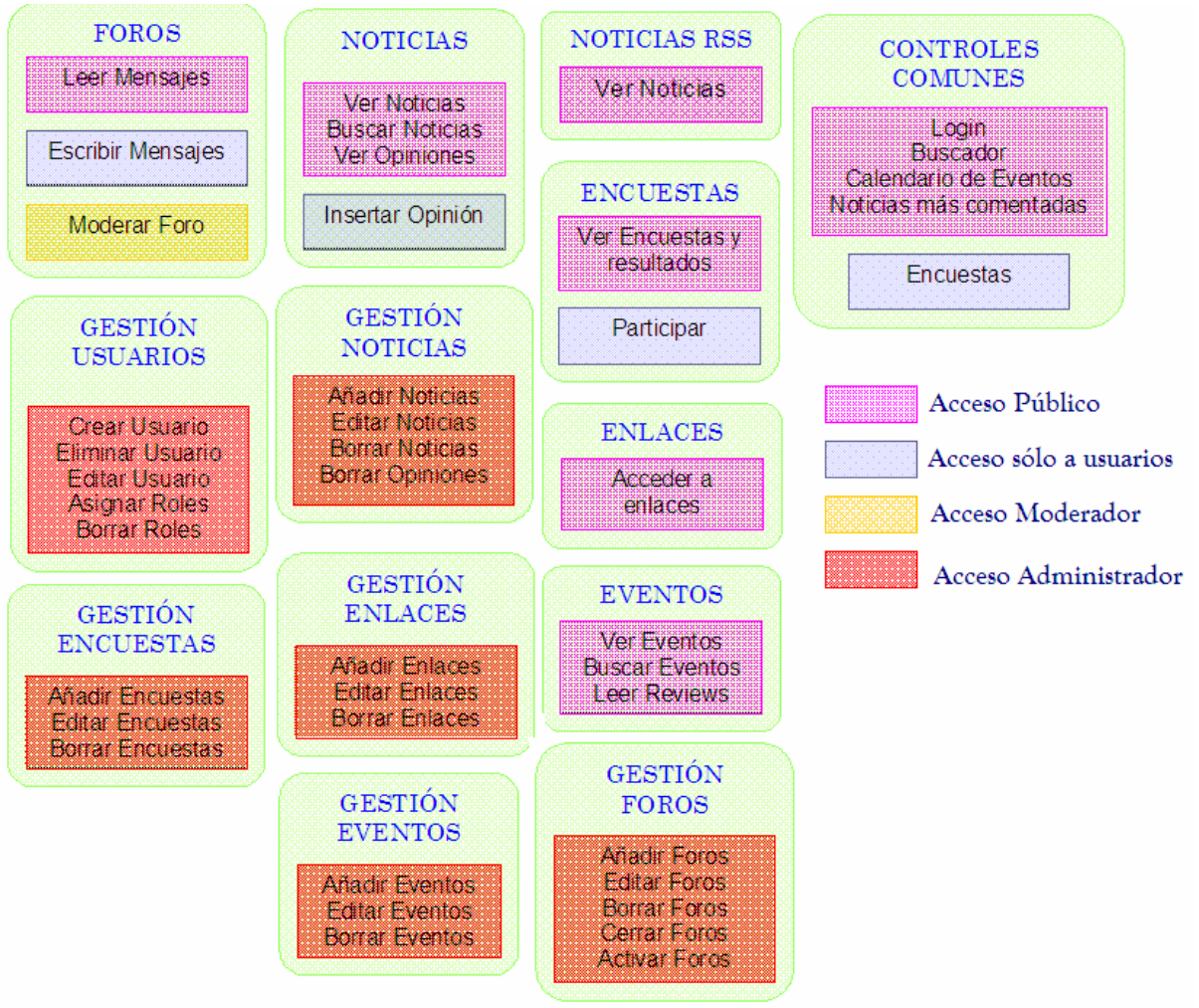


Fig. 95 Control de Acceso del Sitio Web.

6.16.2. Definición de Roles

Es importante, a la hora de regular el acceso a los contenidos del sitio Web, el diseño de una serie de perfiles o roles capaz de ajustarse a las necesidades de cada tipo de usuario. En esta labor, se debe tener en cuenta qué contenidos queremos que estén al alcance de los usuarios, y cuales de ellos creemos conveniente que presenten algún tipo de restricción o regulación.

Asimismo, se debe también contemplar la perspectiva de quién se encargará del mantenimiento, gestión y administración del sitio Web. Todo ello irá encaminado a favorecer una correcta gestión de contenidos y permisos, así como a mejorar en todo lo posible los aspectos relacionados con la seguridad y privacidad.

Existirán diversos roles: el de usuarios anónimos, el de usuarios registrados, el de moderador, y finalmente el de administrador. A continuación pasaremos a describir cada uno de ellos, detallando cuáles son sus funciones y a qué elementos o contenidos podrán acceder. Haremos un especial hincapié en el último de ellos, por su envergadura, y por el importante papel que desempeñará en nuestro sitio.

6.16.2.1. Usuario anónimo

El primero de estos roles, y el de mayor sencillez, es el de acceso anónimo. Se considera usuario anónimo a todo aquel visitante del sitio que no se ha autenticado previamente. Como es lógico, no dispondrá de acceso a todos los contenidos del sitio, pero sí a la mayoría de ellos, sobre todo a aquellos que puedan ser empleados en “modo lectura”. La intención será favorecer el acceso generalizado al sitio, y la de no caer en la tentación de restringir masivamente el acceso a los contenidos, para crear así una sensación de comodidad y libertad por parte de los visitantes.

Por otro lado, cuando un usuario anónimo intente acceder a algún recurso al que no está autorizado, se le invitará a registrarse para así poder ser partícipe de ese contenido, con lo que facilitaremos su rápido y sencillo registro.

A continuación se enumeran las facultades de los usuarios anónimos.

- Acceso a los controles de:
    - Login
    - Buscador
    - Calendario de eventos
    - Noticias más comentadas.
    - Encuestas (lectura).

- Acceso a las páginas de:
    - Noticias: lectura de noticias y opiniones.
    - Portada: lectura de últimas noticias, acceso a buscador de noticias anteriores.
    - Noticias vía RSS: lectura de noticias.
    - Enlace: Acceso a los enlaces.

- Foro: lectura de posts.
- Eventos: lectura de contenidos, reviews, búsquedas…

6.16.2.2. Usuario registrado

El otro rol existente, de lado del cliente es el de usuario registrado. Se trata de un rol que surge de la necesidad de regular el acceso a ciertos controles y contenidos, y cuya misión principal es la de controlar los mecanismos de participación de nuestro sitio Web.

La posibilidad de poder opinar acerca de una noticia o un disco, o de emitir una votación acerca de una encuesta, requiere que su manejo sea sometido a cierto control para favorecer así su correcto uso y funcionamiento. Para ello, imprimiremos reglas de acceso a recursos como los foros, los espacios de opinión, o en definitiva, todos aquellos en los que se produzca una interacción de doble sentido entre el usuario, y los contenidos.

A continuación se detallan los recursos accesibles a un usuario previamente registrado.

- Acceso a los controles de:
  - Login
  - Buscador
  - Enlaces
  - Calendario de eventos
  - Noticias más comentadas.
  - Encuestas (participación)

El acceso a las páginas es el mismo que el usuario anónimo, pero esta vez con la capacidad de poder emitir opiniones y leer o escribir en los Foros.

6.16.2.3. Usuario moderador

El rol de moderador es un rol pensado única y exclusivamente para desempeñar pequeñas labores de gestión en el apartado de foros. Los moderadores serán designados por parte del

administrador, y tendrán además todas las facultades de las que dispone cualquier usuario registrado. Las tareas específicas que podrán desempeñar son:

- Crear nuevos temas/mensajes.
- Decidir que mensajes permanecerán o no visibles.

La finalidad última del moderador es velar por el buen funcionamiento del foro, y controlar y regular las participaciones de los usuarios.

6.16.2.4. Usuario administrador

Se trata de un modo indispensable en todo sitio Web con el que aquellas personas encargadas de mantener el portal, realizarán labores de gestión y administración de manera remota.

Se accede a él sólo si pertenecemos al grupo o rol de administradores. Distinguimos varias secciones dentro del modo administrador: gestión de usuarios, gestión de noticias, gestión de enlaces, gestión de encuestas, gestión de eventos, gestión de foros y gestión de canales RSS. Cada uno de ellos será objeto un exhaustivo análisis a continuación.

6.17. Modo Administrador

6.17.1. Gestión de Noticias

6.17.1.1. Descripción

El modo de gestión de noticias permitirá editar y modificar todas las noticias publicadas en el portal, así como añadir nuevas para que pasen a formar parte de la base de datos. Este modo se compone de dos páginas GestionNoticias.aspx y AddEditNoticia.aspx. En la primera de ellas aparecerá un listado paginado con todas y cada una de ellas, junto con tres opciones en uno de sus márgenes: opiniones, editar y eliminar. Y en la segunda se podrán añadir o editar las noticias.

GESTIÓN DE NOTICIAS

NOTICIAS EN LA BASE DE DATOS

Título	Fecha			
Una oleada de correos fraudulentos intenta obtener claves de usuarios de Telefónica	23/06/2008	Opiniones	Editar	Eliminar
Kaspersky Lab encuentra una posible solución para recuperar los datos cifrados por Gpcode.ak	23/06/2008	Opiniones	Editar	Eliminar
El iPhone SDK alcanza las 250.000 descargas	23/06/2008	Opiniones	Editar	Eliminar
Fuente Online	22/06/2008	Opiniones	Editar	Eliminar
Iberoamérica busca estrategias para reducir brecha digital	18/06/2008	Opiniones	Editar	Eliminar

1 2

Agregar Nueva Noticia

OPINIONES SOBRE LA NOTICIA SELECCIONADA

Usuario	Opinión	Fecha		
admin	Se debe tener mucho cuidado con nuestras contraseñas	23/06/2008 p.m.	08:18:00	Eliminar

Fig. 96 Página Gestión de Noticias.

Si se hace clic en opiniones, se desplegarán todas las opiniones relacionadas con la noticia seleccionada. A su vez, se tendrá la posibilidad de eliminar cualquiera de éstas con tan sólo hacer clic en el botón eliminar, situado a la derecha de cada una de ellas.

La existencia de este control es necesario para poder tener cierto control sobre los comentarios que los usuarios puedan llegar a realizar sobre las noticias, y asegurar así que en ningún caso se están sobrepasando ciertos límites en cuanto al los contenidos de los mismos.

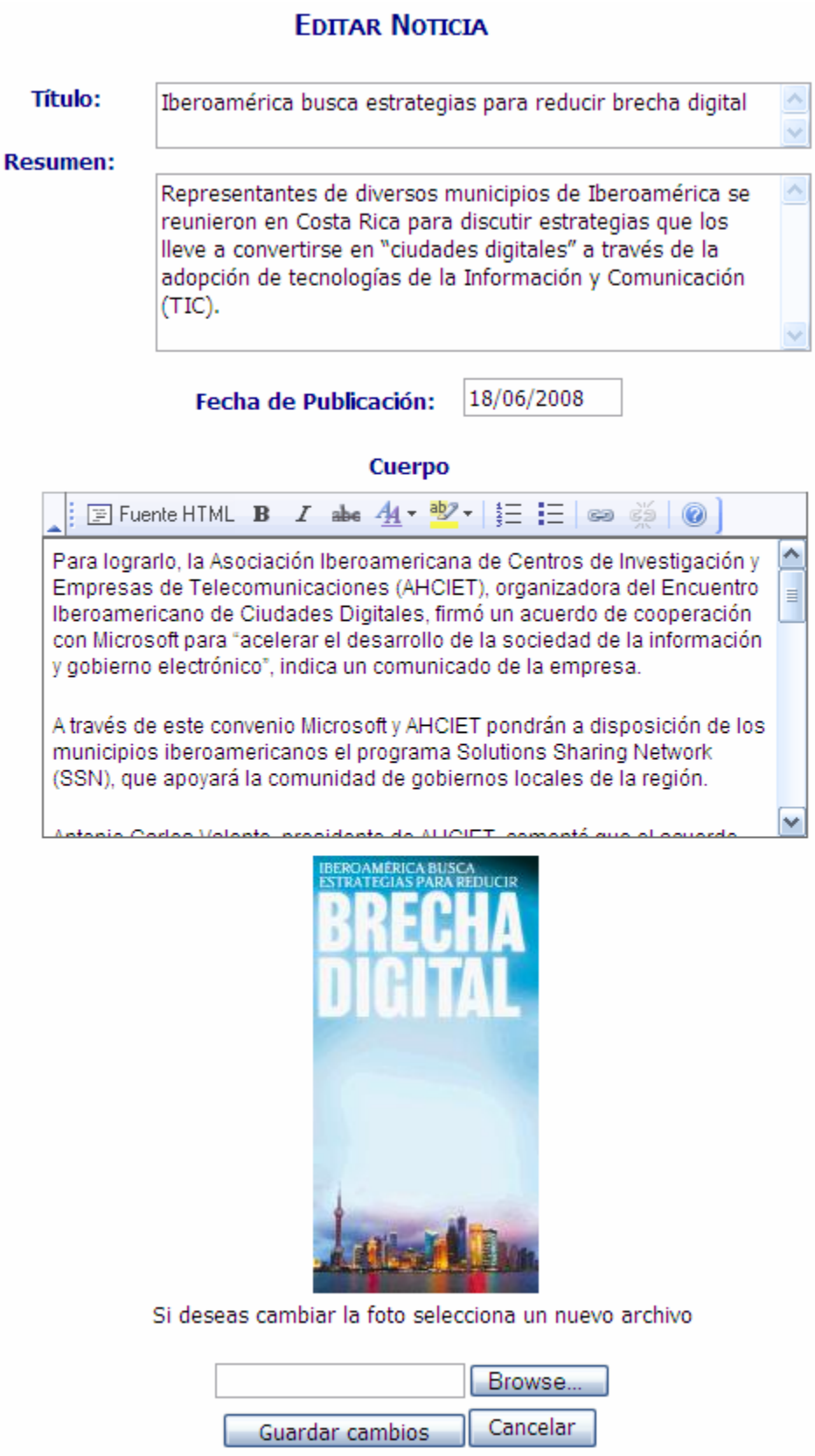


Fig. 97 Editar Noticias.

Si se hace clic en la opción Editar, será redirigido a la página AddEditNoticia.aspx donde se verán todos los detalles de la noticia seleccionada: título, resumen, cuerpo, foto y fecha. Pudiendo pues, modificar cada uno de sus campos, e incluso, reemplazar la foto asociada existente. Además, contamos con la ayuda de un editor de texto (Fckeditor), que nos facilitará la labor de redactar y dar formato al cuerpo de la información.

Una vez terminada la edición de la noticia, bastará con hacer clic en el botón Guardar cambios para actualizar el registro de la base de datos. Asimismo, también se podrá crear nuevas noticias. Para ello debe acudir a la opción Agregar nueva noticia, situada justo debajo del listado de noticias existentes en la base de datos. Al seleccionar dicha opción seremos redirigidos de igual forma que al editar una noticia a la página AddEditNoticia.aspx, con la salvedad de que todos los campos de los formularios estarán vacíos. Al crear una nueva noticia, el sistema le adjudicará un nuevo Id (autoincrementable) y pasará a formar parte de la colección de noticias.

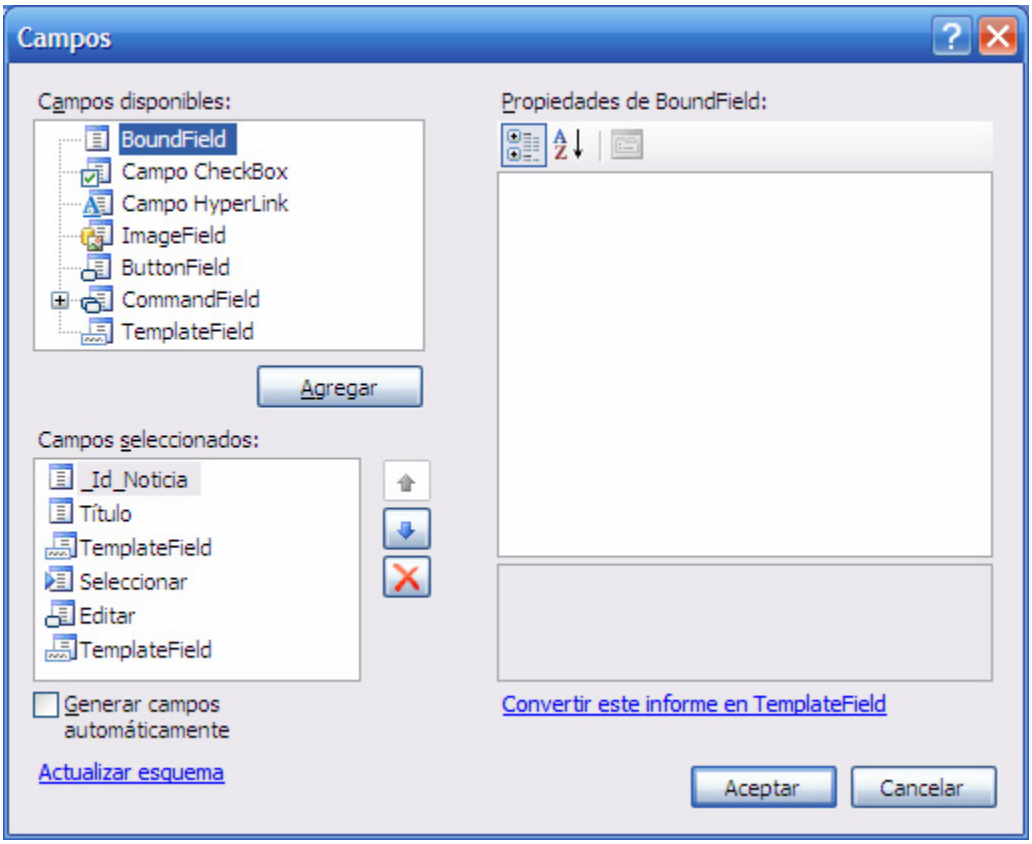
6.17.1.2. Página GestiónNoticias.aspx

Todas las páginas relacionadas con la administración del Portal, serán añadidas en una carpeta (Editores) que cuelga de la raíz del sitio Web. Al igual que todas las páginas del sitio, esta página hará uso de un panel para colocar en él los controles que formarán la misma. En primer lugar añada tres etiquetas, una para dar un título a la página, otra para indicar las noticias y otra para las opiniones relacionadas con las noticias. Además agregue un HyperLink cuya propiedad Text indicará la opción de “Agregar Nueva Noticia”, la dirección de postback será la página AddEditNoticia.aspx.

Seguidamente sitúe un ObjectDataSource para extraer la información correspondiente con las noticias almacenadas en la base de datos (odsNoticias).

```
<asp:ObjectDataSource ID="odsNoticias" runat="server"
DataObjectTypeName="PortalWeb.BO.Noticia"
        DeleteMethod="Delete" OldValuesParameterFormatString="original_{0}"
SelectMethod="GetNoticiasPaginadas"
        TypeName="PortalWeb.BLL.BLLNoticia" UpdateMethod="AddUpdate"
        SelectCountMethod="GetNumNoticias" EnablePaging="True" >
</asp:ObjectDataSource>
```

A continuación agregue un UpdatePanel y en su interior un control GridView (gvNoticias) cuyo origen de datos será dicho ObjectDataSource. Establezca como campo clave del control el campo _Id_Noticia y seguidamente pase a editar las columnas del mismo.



Se tendrán seis columnas, la primera relacionada con el campo _Id_Noticia, aunque no será visible. La segunda la relacionaremos con el campo _Título. Mientras que la tercera columna será un campo del tipo TemplateField, en cuya plantilla ItemTemplate colocaremos una etiqueta relacionada con el campo _Fecha.

Por otro lado habilite la selección, y sitúe dicha columna en cuarto lugar estableciendo su propiedad SelectText igual a “Opiniones”. También añadiremos un campo del tipo Button cuya propiedad CommandName seré igual a Edit y su propiedad Text a “Editar”. Por último añadiremos un campo TemplateField y en su plantilla ItemTemplate agregaremos un LinkButton con las siguientes propiedades:

```
<asp:TemplateField><ItemTemplate>
<asp:LinkButton id="LinkButton1" runat="server" __designer:wfdid="w27"
CommandName="Delete" OnClientClick="return confirm ('Estas seguro de querer borrar
esta noticia?');" CausesValidation="False">Eliminar</asp:LinkButton>
</ItemTemplate>
</asp:TemplateField>
```

Este botón pedirá confirmación del usuario para posteriormente eliminar la noticia seleccionada.

Por otro lado tendremos añada un manejador para el evento RowCommand del GridView, y cuyo código será el siguiente:

```
protected void gvNoticias_RowCommand(object sender, GridViewCommandEventArgs e)
   {
      switch (e.CommandName.ToLower())
      {
         case "edit":
            int indicefila = Convert.ToInt32(e.CommandArgument);
            int noticiaid = Convert.ToInt32(gvNoticias.DataKeys[indicefila].Value);

Response.Redirect(String.Format("~/Editores/AddEditNoticia.aspx?idadm=1&id={0}",
noticiaid.ToString()));
            break;
      }
   }
```

De tal manera hacemos que cuando pulsemos el botón editar seamos redirigidos a la página AddEditNoticia, pasando como parámetro QueryString el identificador de la noticia que queremos editar. Asimismo habilite la opción de paginación del control GridView (AllowPaging) a true.

Una vez configurado el GridView gvNoticias, añada otro ObjectDataSource (odsOpiniones) que será el encargado de recoger la información referente a las opiniones relacionadas con una determinada noticia. Configuraremos dicho control del siguiente modo:

```
<asp:ObjectDataSource ID="odsOpiniones" runat="server"
DataObjectTypeName="PortalWeb.BO.Opinión"
      DeleteMethod="Delete" EnablePaging="True"
MaximumRowsParameterName="pageSize"
      OldValuesParameterFormatString="original_{0}"
SelectCountMethod="NumOpinionesByIdNoticia"
      SelectMethod="ListByIdNoticiasPaginadas"
StartRowIndexParameterName="pageIndex"
```

```
      TypeName="PortalWeb.BLL.BLLOpinión" UpdateMethod="AddUpdate">
      <SelectParameters>
        <asp:ControlParameter ControlID="gvNoticias" Name="id"
PropertyName="SelectedValue"
            Type="Int32" />
      </SelectParameters>
    </asp:ObjectDataSource>
```

De nuevo se hace uso de la técnica de paginación Custom Paging, por lo que como método asociado a la operación SELECT establecemos el método ListByIdNoticiasPaginadas. Dicho método recibe como parámetro el identificador correspondiente a la noticia, el origen del mismo será el valor seleccionado en el GridView gvNoticias.

Lo siguiente será añadir un UpdatePanel y en su interior otro GridView, gvOpiniones. En este caso, el campo clave será _Id_Opinión y los campos a mostrar serán el nombre del usuario, la opinión y la fecha en la que fue realizada. Para posibilitar la opción de eliminar opiniones añadiremos el mismo campo del tipo Template Field en cuyo interior situaremos un LinkButton configurado de igual forma que en el caso anterior. Al igual que en el GridView anterior, habilite la paginación.

Añada un método para el manejador PageIndexChaged del GridView correspondiente a las noticias, para que cuando cambiemos de página la noticia seleccionada siempre sea la primera.

```
protected void gvNoticias_PageIndexChanged(object sender, EventArgs e)
    {
        gvNoticias.SelectedIndex = 0;
    }
```

Finalmente edite los controladores de los eventos RowCommand, de ambos GridView:

```
protected void gvNoticias_RowDeleted(object sender, GridViewDeletedEventArgs e)
    {
        //Si borramos la última fila del GridView, decrementamos el PAgeIndex
        if (e.Exception == null && gvNoticias.Rows.Count == 1)
        {
            // borramos la última fila
            gvNoticias.PageIndex = Math.Max(0, gvNoticias.PageIndex - 1);
        }
    }

    protected void gvOpiniones_RowDeleted(object sender, GridViewDeletedEventArgs e)
    {
```

```
        if (e.Exception == null && gvOpiniones.Rows.Count == 1)
                gvOpiniones.PageIndex = Math.Max(0, gvOpiniones.PageIndex - 1);
    }
```

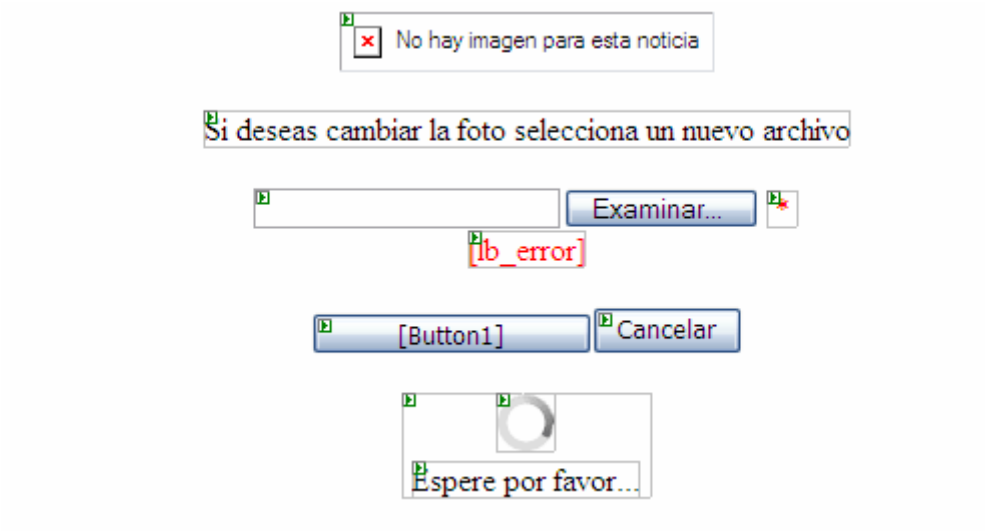
6.17.1.3. Página AddEditNoticia.aspx

La página AddEditNoticia.aspx estará formada por diversos controles que se encontrarán situados dentro de un gran panel. En primer lugar sitúe un Label que albergará el nombre de la página. A continuación inserte una tabla formada por 2 filas y 2 columnas, destinada a dar cabida a la información de los campos Título y Resumen. En la columna de la izquierda ubique un par de etiquetas con el nombre de dichos campos, y en la de la derecha, dos controles de tipo TextBox. Junto a cada uno de ellos colocaremos controles de tipo RequiredFieldValidator que alertarán en caso de que dichos campos permanezcan vacíos en el momento de guardar los datos de la página, o también cuando los campos contengan caracteres inválidos o superen cierto número de caracteres. También utilice los controles RegularExpressionValidator, para validar una expresión de entrada, recuerde colocar todos estos controles con la misma propiedad ValidationGroup.

Para el campo fecha, ubicado inmediatamente debajo de la tabla, emplee también un control TextBox junto con su etiqueta correspondiente.

A continuación, y ya para el cuerpo de la noticia, utilice el editor de textos FCkEditor. Con él podrá editar y modificar el texto principal de la noticia. Al tratarse de un campo obligatorio, le asociaremos también un RequiredFieldValidator.

En cuanto a la imagen asociada a la noticia, para su visualización o inserción, se recurre a un conjunto de controles. Por un lado, inserte un control Image para visualizar la foto actual de dicha noticia. En caso de que se desee insertar una nueva, deberá hacer uso de un control FileUpload, ubicado debajo de una Label que nos los indicará. Haciendo clic en el botón “Examinar” podrá elegir el archivo de imagen que deseamos subir. Junto a él coloque otro control RequiredFieldValidator para alertarnos de su obligatoriedad.



Por último, coloque un control UpdateProgress para escenificar de forma gráfica el proceso de actualización del UpdatePanel sobre el que han sido colocados todos los elementos de la página.

Ahora bien, la página AddEditNoticia será la misma tanto para el caso en el que se vaya a editar una noticia ya existente, como para el caso en el que se vaya a insertar una nueva. Personalizaremos la misma en función de si su finalidad es una u otra, para que los campos aparezcan llenos o vacíos. Y el indicador que nos permitirá discernir entre ambas no será otro que el _Id_Noticia. En el siguiente código se observa como realizar estas acciones.

```
public partial class Editores_AddEditNoticia : BasePage
{
    int idnoticia;
    bool edit = false;

    protected void Page_Load(object sender, EventArgs e)
    {
        if (Request.QueryString.Get("id") != null)
            idnoticia = Convert.ToInt32(Request.QueryString.Get("id"));

        if (!Page.IsPostBack)
        {
            if (idnoticia > 0)
            {
                Noticia minoticia = BLLNoticia.SelectGetById(idnoticia);
                if (minoticia != null)
                {
                    ctTitulo.Text = minoticia._Título;
                    ctResumen.Text = minoticia._Resumen;
```

```
                FCKeditor1.Value = minoticia._Cuerpo;
                ctFecha.Text = minoticia._Fecha.ToShortDateString();
                Image1.ImageUrl = minoticia._Imagen;
                this.Title = "Editar Noticia";
                etCabecera.Text = "Editar Noticia";
                Button1.Text = "Guardar cambios";
                edit = true;
                rfvFoto.Enabled = false;
            }
        }
        else //Creamos uno nuevo insert
        {
            ctFecha.Text = DateTime.Now.ToShortDateString();
            this.Title = "Nueva noticia";
            etCabecera.Text = "Añadir noticia";
            Button1.Text = "Agregar noticia";
        }
    }
}
```

Cómo se puede apreciar, en primer lugar, acudiremos a la QueryString para averiguar cual es el valor del id pasado. Cuando éste tenga un valor mayor que 0, significará que estamos tratando con una noticia ya existente, es decir, la operación a realizar será “Editar noticia”. En caso contrario, estaremos insertando una noticia nueva.

En el primero de los casos se deberá acudir al método SelectGetById de la clase BLLNoticia, al que se le pasará por valor el identificador de la noticia que pretendemos obtener. Dicho método nos devolverá el objeto Noticia correspondiente, cuyos campos asociaremos con todos los controles anteriormente insertados.

Por otro lado, personalizaremos también algunos de los botones y etiquetas, como por ejemplo el texto de cabecera de la página o el botón final de “Guardar cambios”.

Ahora bien, en caso de que se trate de una noticia nueva, dejaremos todos los controles vacíos, salvo el de la fecha, en el que aparecerá por defecto el día actual. También cambiará el texto mostrado por algunas etiquetas o botones. Cuando se haga clic en el botón final, destinado a guardar la información, acudiremos al manejador del evento clic que tendrá el siguiente código.

```
protected void Guardar_Cambios(object sender, EventArgs e)
   {
      try
      {
         Noticia minoticia;
         if (idnoticia > 0)
            minoticia = BLLNoticia.SelectGetById(idnoticia);
         else //noticia nueva, fecha nueva.
         {
            minoticia = new Noticia();
            minoticia._Fecha = DateTime.Now;
         }
         minoticia._Título  = this.ctTitulo .Text;
         minoticia._Resumen = this.ctResumen.Text;
         minoticia._Cuerpo = FCKeditor1.Value;
         if (FileUpload1.HasFile)
         {
            FileInfo infofoto = new FileInfo(FileUpload1.PostedFile.FileName);
            if (((infofoto.Extension.ToLower() == ".jpg") || (infofoto.Extension.ToLower() ==
".jpeg"))
                  && (FileUpload1.FileContent.Length < 100 * 1024))
            {
               if (!edit)
               {
                  minoticia._Imagen = " ";
                  idnoticia = BLLNoticia.AddUpdate(minoticia);
               }
FileUpload1.SaveAs("C:\\Inetpub\\wwwroot\\PortalWeb\\Controles\\Noticias\\Fotos\\" +
infofoto.Name);

               string nombrefoto = "~/Controles/Noticias/Fotos/" + infofoto.Name;
               minoticia._Imagen = nombrefoto;
               BLLNoticia.AddUpdate(minoticia);
            }
            else
               throw new Exception("La foto no es tipo jpg o supera los 100Kbytes");
        }
         else
              BLLNoticia.AddUpdate(minoticia);
         Panel2.Visible = true;
         Response.Redirect("~/Editores/GestionNoticias.aspx?idadm=1");
      }
      catch (Exception exp)
      {
         lb_error.Text = exp.Message;
      }
}
```

En primer lugar se compruebe si se trata de una ya existente o de una nueva. En el primer caso, se acude al método SelectGetById para obtener la noticia correspondiente, y en el segundo se crea un nuevo objeto de tipoNoticia, cuyo campo fecha inicializaremos desde un

primer momento. Acto seguido, y para ambas situaciones, rellenaremos los campos _Título, _Resumen y _Cuerpo.

Para el caso de la imagen, se actúa de forma diferente. Primeramente se comprueba si el control FileUpload contiene algún archivo, es decir, se ha modificado la foto existente o se ha insertado una nueva. Acto seguido se verifican algunos parámetros de la imagen, como su formato o su tamaño. Ésta habrá de ser de tipo jpg y no superar los 100 kb. En el caso de que todo transcurra correctamente, se llama al método AddUpdate de la clase BLLNoticia para obtener el id de la noticia con la que estamos tratando. En caso contrario se lanzará una excepción que avisará de que no hemos cumplido con alguno de los requisitos.

Tras hacer clic en el botón de guardar cambios se nos remitirá automáticamente a la página principal de gestión de noticias y haremos visible el pequeño panel en el que está ubicado el control UpdateProgress. Lo mismo ocurrirá si hacemos clic en el botón cancelar.

```
protected void cancelar_click(object sender, EventArgs e)
    {
        Panel2.Visible = true;
        Response.Redirect("~/Editores/GestionNoticias.aspx?idadm=1");
    }
```

6.17.2. Gestión de Eventos

6.17.2.1. Descripción

En esta sección se podrá administrar y gestionar todo lo relacionado con los eventos que van a tener cabida en la Web.

En primer término aparecerá un listado paginado de todos los eventos existentes, ordenados de la más antigua a la más lejana. En este listado figurarán tres datos fundamentales: el nombre del evento, la ciudad donde tendrá lugar el evento, y la fecha del acontecimiento. Junto a ellos, un par de botones: editar y eliminar. Si hacemos clic en el botón eliminar, suprimiremos directamente dicho evento.

Fig. 98 Gestión de Eventos.

Si por el contrario queremos editar alguno de sus datos, seleccionaremos la opción editar. Se nos desplegará ahora una nueva página con todos los detalles del evento seleccionado.

Se podrá editar el título del evento, la ciudad, la descripción, la página de enlace, la fecha del evento, y también la imagen asociada. Para la selección de la fecha contaremos con la ayuda de un calendario desplegable, sobre el que podremos seleccionar la fecha que deseemos con tan sólo hacer clic en el día indicado. La foto la elegiremos a través de un control FilleUpload. Por último, también podremos insertar una crónica del evento tras su celebración, mediante el editor de texto Fckeditor del campo review.

Ahora bien, si lo que se pretende es añadir un nuevo evento, habrá de hacer clic en el botón “Agregar nuevo Evento” ubicado en la parte inferior de la página principal de Gestión de eventos, para que se nos despliegue una página idéntica a la mostrada para la opción editar, con la única salvedad de que en esta ocasión todos los campos aparecerán en blanco.

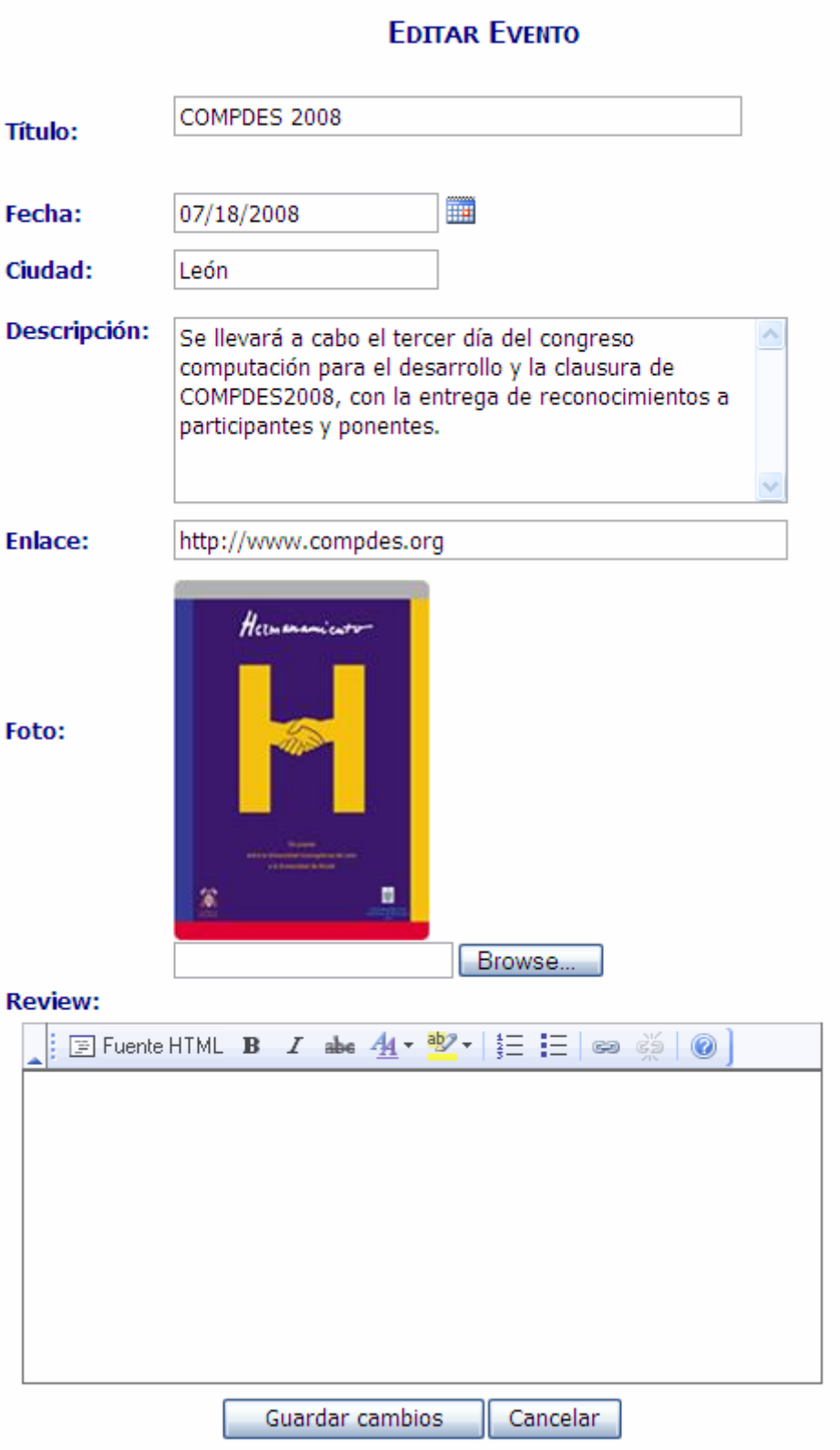


Fig. 99 Editar un Evento.

6.17.2.2. Página GestiónEventos.aspx

El elemento principal de la página GestionConciertos.aspx será un GridView similar a los existentes en el modo de gestión de noticias. Irá ubicado en su correspondiente panel, y en la parte superior contará con una etiqueta indicando el tipo de modo de gestión, y en la parte inferior un enlace para agregar nuevos eventos.

En cuanto al enlace de datos, dispondremos de un ObjectDataSource conectado a los correspondientes métodos de la clase BLLEvento.

```
<asp:ObjectDataSource ID="odsEventos" runat="server"
DataObjectTypeName="PortalWeb.BO.Evento"
        DeleteMethod="Delete" EnablePaging="True"
OldValuesParameterFormatString="original_{0}"
        SelectCountMethod="GetNumEventosPaginados"
SelectMethod="GetEventosPaginados"
        TypeName="PortalWeb.BLL.BLLEvento" UpdateMethod="AddUpdate">
 </asp:ObjectDataSource>
```

Por lo que respecta al GridView, éste contará con una serie de columnas en las que se detallarán tan sólo algunos de los campos del objeto Evento. Será el caso del campo Título, Ciudad, y Fecha. También añadiremos un campo del tipo Button cuya propiedad CommandName será igual a Edit y su propiedad Text a “Editar”. Por último añadiremos un campo TemplateField y en su plantilla ItemTemplate agregaremos un LinkButton que permita eliminar un evento, igual que la eliminación de una Noticia.

El manejador para el evento RowCommand del GridView nos permitirá ser redirigidos a la página AddEditEvento al pulsar el botón editar, pasando como parámetro QueryString el identificador del evento que queremos editar. Edítelo siguiendo como ejemplo la página GestiónNoticias.aspx.

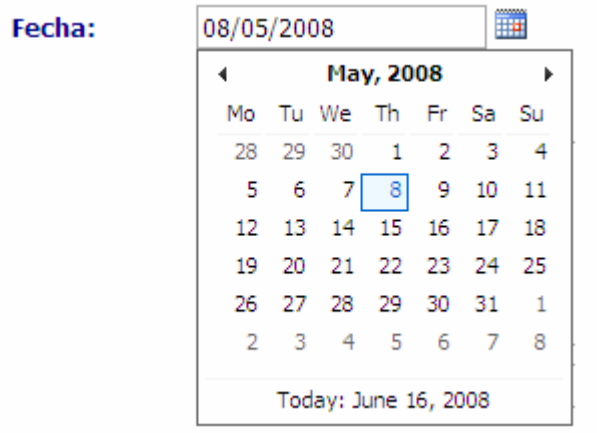
Recuerde habilitar la paginación del GridView y editar el controlador del evento RowDeleted.

6.17.2.3. Página AddEditEventos.aspx

La página AddEditEvento.aspx estará formada por diversos controles que se encontrarán situados dentro de un gran panel. En primer lugar situaremos una Label con el nombre de la página. A continuación insertaremos una tabla formada por 2 columnas y múltiples filas, en las que iremos insertando los controles correspondientes para cada uno de los campos del objeto Evento.

Nos encontraremos pues con numerosos controles Textbox para introducir los datos, así como con sus etiquetas asociadas. También incluiremos los controles RequieredFieldValidator correspondientes, que nos alertarán en caso de que dichos campos permanezcan vacíos en el momento de guardar los datos de la página, o también cuando los campos contengan caracteres inválidos o superen cierto número de caracteres.

Para el campo fecha contamos con la novedad de la inclusión de un control Calendar Extender. Éste puede asociarse con cualquier control TextBox y permite que el cliente pueda interactuar con un pequeño calendario a la hora de seleccionar una determinada fecha. Además, las flechas de desplazamiento permiten seleccionar un mes posterior o anterior al actual.




```
<ajaxToolkit:CalendarExtender ID="CalendarExtender1" runat="server"
FirstDayOfWeek="Monday"
                    Format="dd/MM/yyyy" PopupButtonID="Image1"
TargetControlID="ctFecha">
  </ajaxToolkit:CalendarExtender>
```

Asimismo, agregue un control extender MaskedEditExtender que también añadirá funcionalidad al control TextBox de la fecha. Cuando lo usamos, la entrada del TextBox está condicionada a una máscara y el valor introducido es validado de acuerdo con el tipo de dato elegido. Verificará si la fecha proporcionada es una fecha valida.

```
<ajaxToolkit:MaskedEditExtender ID="MaskedEditExtender1" runat="server"
Mask="99/99/9999"
                    MaskType="Date" TargetControlID="ctFecha">
                </ajaxToolkit:MaskedEditExtender>
```

En cuanto al resto de controles, dispondremos, como en la gestión de Noticias, del editor de textos FckEditor, y también de un control FileUpload para añadir las fotos asociadas a los Eventos.

Edite los controladores del evento Load, y del evento clic del botón para guardar los cambios, utilice como guía la página AddEditNoticias.aspx.

6.17.3. Gestión de Enlaces

6.17.3.1. Descripción

El modo de gestión de enlaces permitirá editar y modificar todos los enlaces publicados en el portal, así como añadir nuevos para que pasen a formar parte de la base de datos y borrar los existentes. Este modo se compone de dos páginas GestionEnlaces.aspx y AddEditEnlace.aspx. En la primera de ellas aparecerá un listado paginado con todos los enlaces, junto con dos opciones en uno de sus márgenes: editar y borrar. Y en la segunda se podrán añadir enlaces.

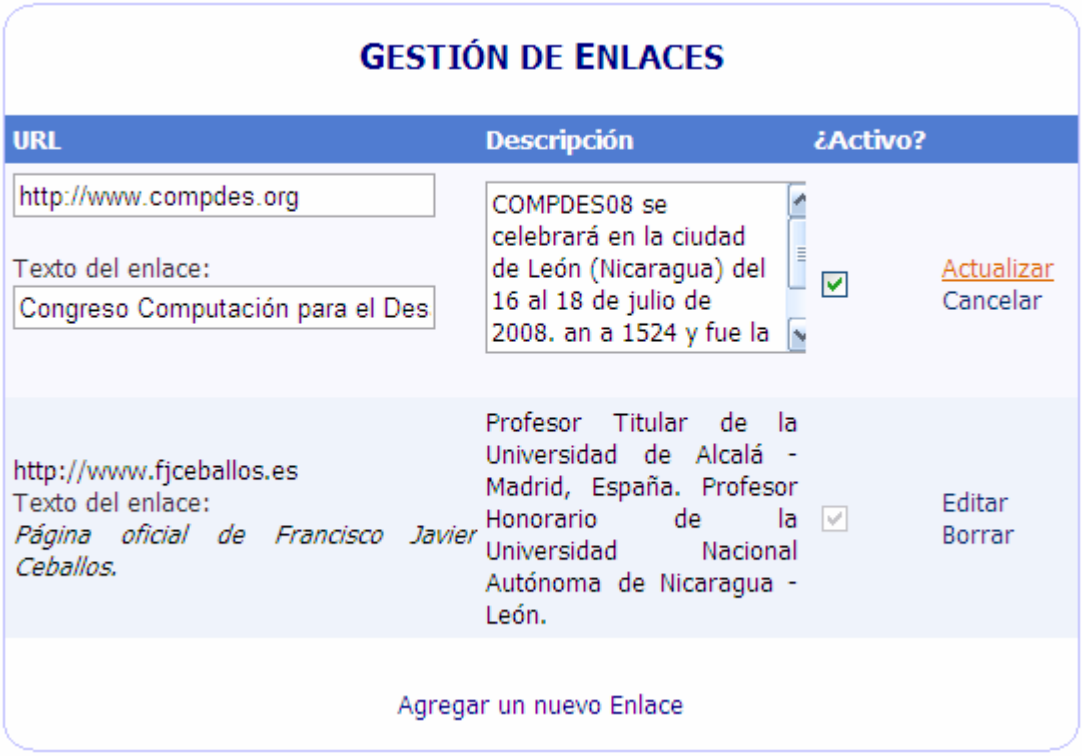


Fig. 100 Gestión de Enlaces.

Si se hace clic en la opción Editar, se colocarán todos los campos que forman el GridView en modo edición, pudiendo de esta manera actualizar los campo de la clase Enlace, con tan sólo dar clic en la opción Actualizar, de lo contrario deberá dar clic en la opción Cancelar, con lo que se cancelan los cambios que realizó sobre las cajas de texto.

Si lo que desea es añadir un nuevo enlace, deberá seleccionar la opción Agregar un nuevo Enlace, predigiriéndose a la página AddEditEnlace.aspx, que le permitirá agregar un nuevo Enlace a la base de datos. Una vez termine de introducir los datos asociados al nuevo Enlace deberá dar clic en la opción Aceptar.

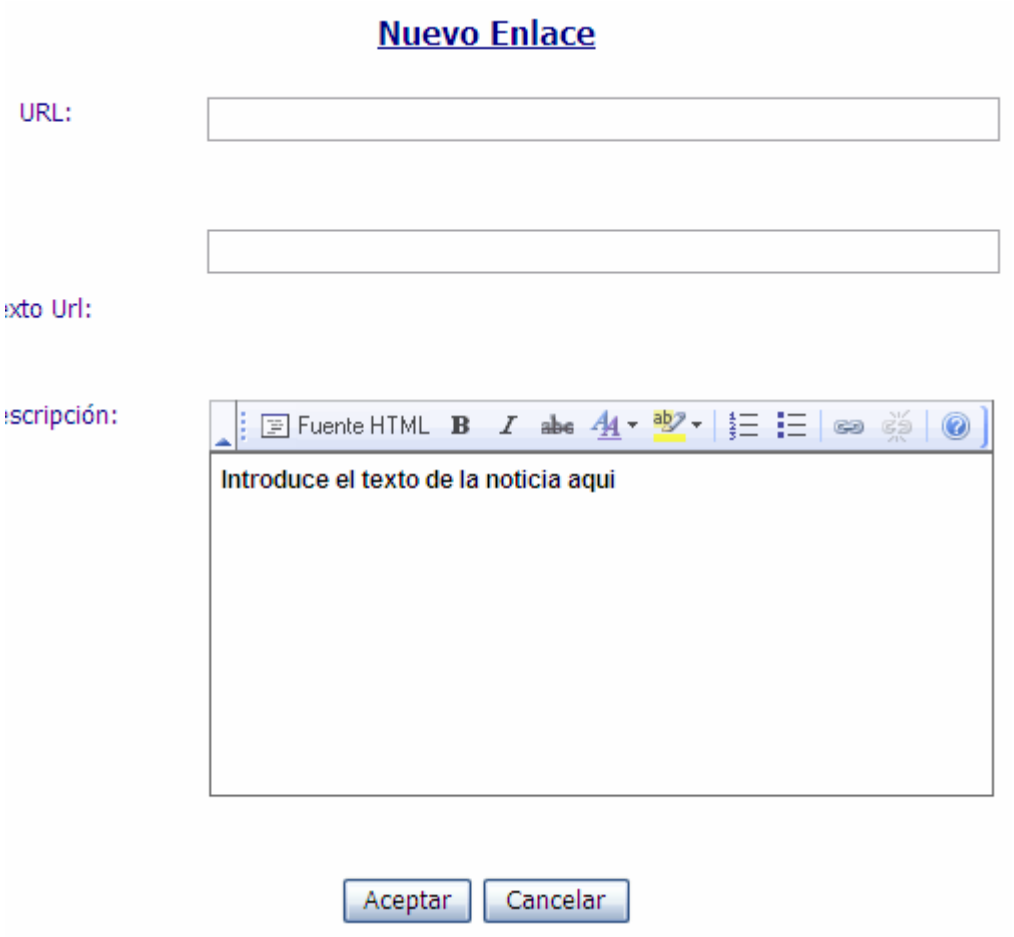


Fig. 101 Agregar un enlace.

6.17.3.2. Página GestiónEnlace.aspx

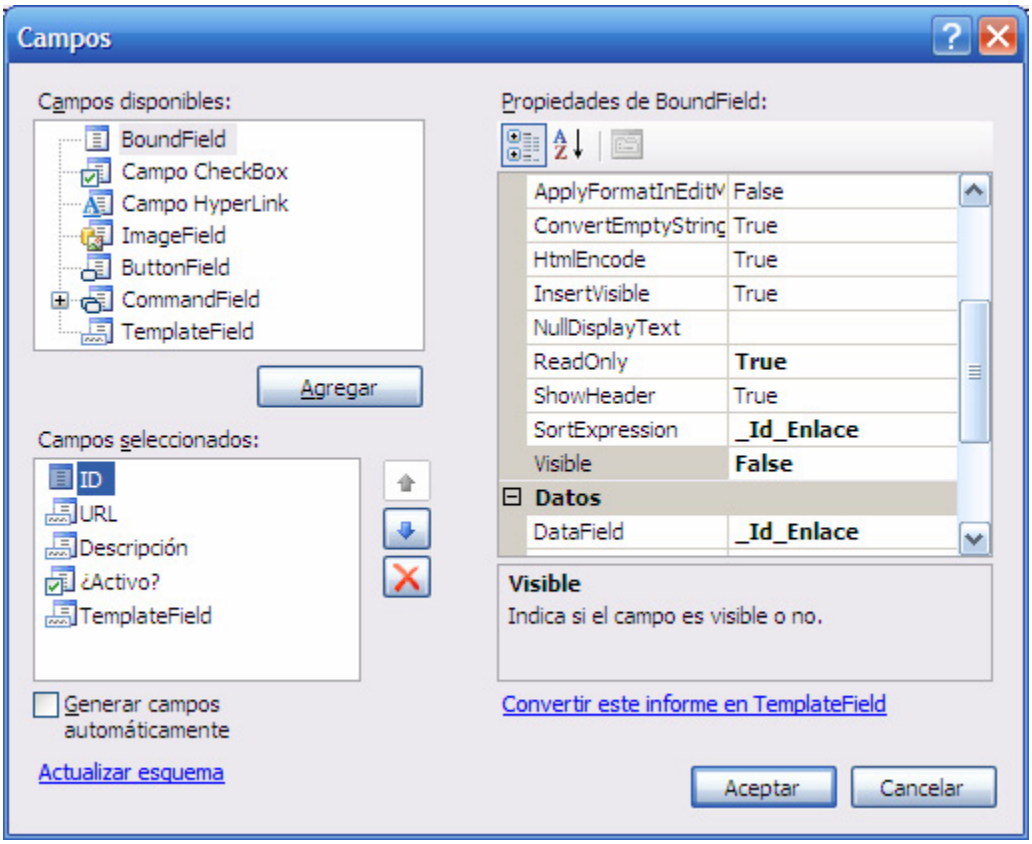
El elemento principal de la página AddEditEnlace.aspx será un GridView similar a los existentes en el modo de gestión de noticias. Irá ubicado en su correspondiente panel y éste dentro de un UpdatePanel, y en la parte superior contará con una etiqueta indicando el título de la página "Gestión Enlace" y en la parte inferior un enlace para agregar nuevos eventos.

En cuanto al enlace de datos, dispondremos de un ObjectDataSource conectado a los correspondientes métodos de la clase BLLEnlace.

```
<asp:ObjectDataSource id="odsEnlaces" runat="server"
TypeName="PortalWeb.BLL.BLLEnlace" DeleteMethod="Delete"
SelectMethod="SelectList" OldValuesParameterFormatString="original_{0}"
DataObjectTypeName="PortalWeb.BO.Enlace"
```

```
UpdateMethod="BLLEnlacesAddUpdate"></asp:ObjectDataSource>
```

En lo que respecta al GridView, éste contará con una serie de columnas en las que se detallarán tan sólo algunos de los campos del objeto Enlace. Será el caso del campo ID que no será visible, URL, Descripción y Activo, cada uno de ellos asociado con su correspondiente campo del objeto Enlace.



También añadiremos un campo del tipo TemplateField y en su plantilla ItemTemplate deberá editar de la siguiente manera:

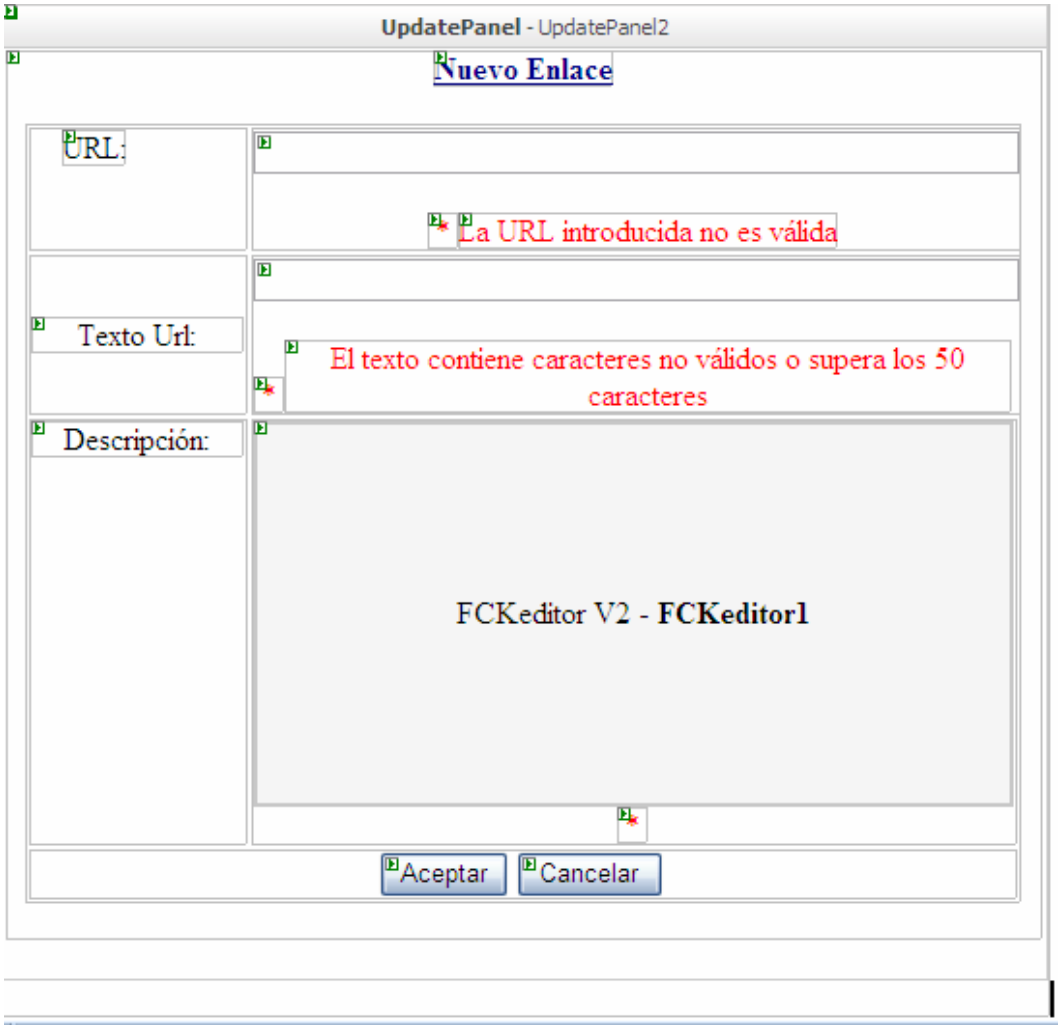
```
<asp:TemplateField><EditItemTemplate>
<asp:LinkButton id="LinkButton1" runat="server" Text="Actualizar" __designer:wfdid="w5"
CommandName="Update" CausesValidation="True"
ValidationGroup="formeditenlace"></asp:LinkButton>  <asp:LinkButton
id="LinkButton2" runat="server" Text="Cancelar" __designer:wfdid="w6"
CommandName="Cancel" CausesValidation="False"
ValidationGroup="formeditenlace"></asp:LinkButton>
</EditItemTemplate>
<ItemTemplate>
<asp:LinkButton id="LinkButton1" runat="server" Text="Editar" __designer:wfdid="w3"
CommandName="Edit" CausesValidation="False"></asp:LinkButton> 
<asp:LinkButton id="LinkButton2" runat="server" Text="Borrar" __designer:wfdid="w4"
```

```
CommandName="Delete" CausesValidation="False" OnClientClick="return
confirm('¿Desea borrar el enlace?')"></asp:LinkButton>
</ItemTemplate>
</asp:TemplateField>
```

Recuerde habilitar la paginación del GridView y editar el controlador del evento RowDeleted.

6.17.3.3. Página AddEditEnlace.aspx

La página AddEditEnlace.aspx estará formada por diversos controles que se encontrarán situados dentro de un gran panel y éste a su vez dentro de un UpdatePanel. En primer lugar situaremos una Label con el nombre “Nuevo Enlace”. A continuación insertaremos una tabla formada por 2 columnas y múltiples filas, en las que iremos insertando los controles correspondientes para cada uno de los campos del objeto Enlace.



Nos encontraremos pues con numerosos controles Textbox para introducir los datos, así como con sus etiquetas asociadas. También incluiremos los controles RequieredFieldValidator correspondientes, que nos alertarán en caso de que dichos campos permanezcan vacíos en el momento de guardar los datos de la página, o también cuando los campos contengan caracteres inválidos o superen cierto número de caracteres.

Para el campo Descripción, se utilizará un control FCKeditor. Al dar clic en el botón Cancelar se redijera a la página GestiónEnlace.aspx, y el controlador del evento clic del botón Aceptar, deberá guardar el nuevo enlace en la base de datos.

```
protected void Button2_Click(object sender, EventArgs e)
    {
      Enlace mienlace = new Enlace();
       mienlace._Activo = true;
       mienlace._Descripción = FCKeditor1.Value;
       mienlace._Url = ctURL.Text;
       mienlace._TextoUrl = ctTextoUrl.Text;
       BLLEnlace.BLLEnlacesAddUpdate(mienlace);
       Response.Redirect("~/Editores/GestionEnlaces.aspx?idadm=1");
    }
```

6.17.4. Gestión de Encuestas

6.17.4.1. Descripción

Para la gestión del control encuestas hemos agregado una sección independiente dentro del modo administrador. Desde ella podremos crear nuevas encuestas, visualizar un historial de las que ya han sido publicadas en nuestro sitio, o editar alguna de sus propiedades.

Si seleccionamos una de las encuestas que nos aparecen en el listado, podremos ver las posibles respuestas existentes. También añadir nuevas respuestas, así como editarlas o eliminarlas.

Fig. 102 Gestión de Encuestas.

Si por el contrario, lo que desea es editar los parámetros de la propia encuesta, hemos de hacer clic en editar. Los valores que podrán ser editados serán, la pregunta, la fecha de inicio y de conclusión del periodo de participación en la encuesta, y por último la demarcación de “activo”, que indicará si se puede participar en la misma estando aún abierto y operativo el plazo para votar.

Fig. 103 Editar Encuesta.

Desde el menú principal también tenemos la opción de crear nuevas encuestas. Simplemente tenemos que especificar la pregunta o tema de la encuesta y su fecha de cierre. La fecha de publicación será la del día en que añadamos la encuesta y por defecto la nueva encuesta estará activa. Una vez pulsemos aceptar, la encuesta aparecerá junto con las restantes en el listado.

Para agregar las diferentes respuestas posibles, seleccionaremos la encuesta del listado y a continuación añadiremos una a una las respuestas.

6.17.4.2. Página GestionEncuesta.aspx

La página de gestión de encuesta estará formada por diversos controles. El primero con el que nos encontramos es un control Gridview. Su origen de datos estará enlazado con el ObjectDataSource, que conectará con el método SelectList() de la clase BLLEncuesta, y que nos devolverá una colección de elementos EncuestaList.

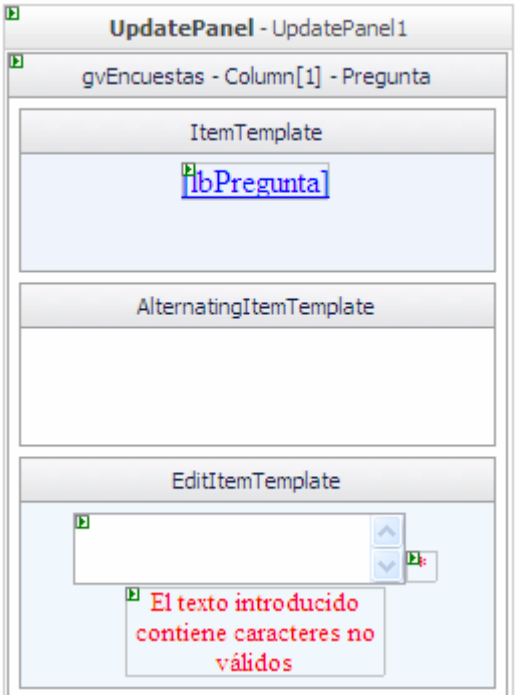
Los campos que mostraremos en cada una de las columnas que conforman el GridView serán: Pregunta, Fecha de publicación, Fecha de finalización y Activa. Convertiremos el campo Pregunta en un campo seleccionable, y lo haremos empleando trasformándolo en un Hyperlink.

```
<ItemTemplate>
<asp:LinkButton id="lbPregunta" runat="server" Text='<%# Eval("_Pregunta") %>'
CssClass="textosleft" __designer:wfdid="w89"
CommandName="Select"></asp:LinkButton>
</ItemTemplate>
```

También añada un campo del tipo LinkButton cuya propiedad CommandName sera igual a Edit y su propiedad Text a “Editar”. Por último añadiremos un campo TemplateField y en su plantilla ItemTemplate agregaremos otro LinkButton para ejecutar la operación Eliminar. Dicho botón pedirá confirmación del usuario para posteriormente eliminar la encuesta seleccionada.

```
<ItemTemplate>
<asp:LinkButton id="LinkButton1" runat="server" __designer:wfdid="w47"
CommandName="Edit" CausesValidation="False">Editar</asp:LinkButton> 
<asp:LinkButton id="LinkButton2" runat="server" __designer:wfdid="w48"
CommandName="Delete" CausesValidation="False" OnClientClick="return
confirm('¿Desea eliminar esta encuesta, así como las respuestas y las votaciones de
usuarios relacionadas?')">Eliminar</asp:LinkButton>
</ItemTemplate>
```

Cuando hagamos clic en el botón Editar, el GridView mostrará una plantilla diferente. Sus campos serán ahora editables, y el aspecto que presentará la primera columna, será el siguiente.

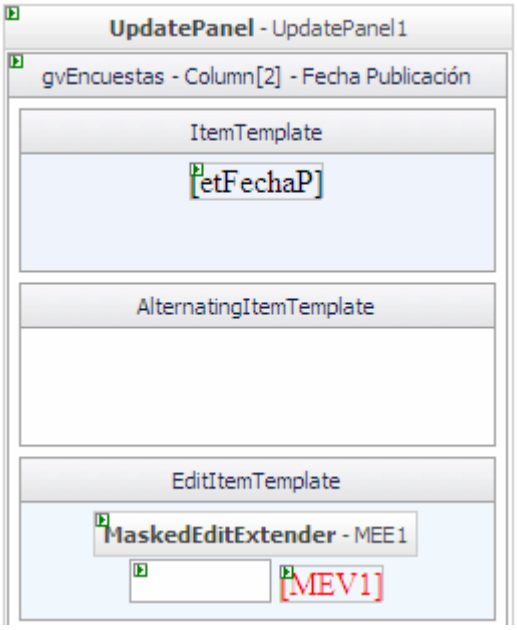


Cuyo código es:

```
<EditItemTemplate >
<asp:TextBox id="TextBox2" runat="server" Width="160px" Text='<%# Bind("_Pregunta")
%>' CssClass="textos" __designer:wfdid="w90" Rows="2" TextMode="MultiLine"
CausesValidation="True"></asp:TextBox><asp:RequiredFieldValidator
id="RequiredFieldValidator4" runat="server" Width="13px" __designer:wfdid="w91"
ValidationGroup="editEncuesta" ErrorMessage="*" ControlToValidate="TextBox2" Font-
Size="Smaller"></asp:RequiredFieldValidator><asp:RegularExpressionValidator
```

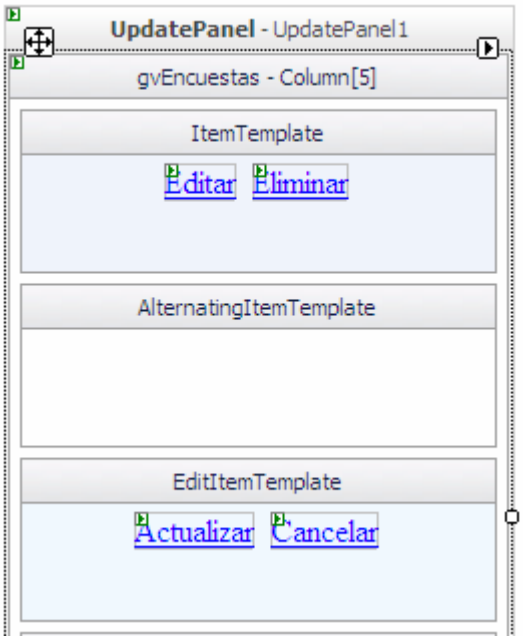
```
id="RegularExpressionValidator1" runat="server" Width="130px" CssClass="textosleft"
__designer:wfdid="w92" ValidationGroup="editEncuesta" ErrorMessage="El texto
introducido contiene caracteres no válidos" ControlToValidate="TextBox2"
ValidationExpression="^[^'%*<>;=]+$" Font-
Size="Smaller"></asp:RegularExpressionValidator>
</EditItemTemplate>
```

Haremos lo mismo con los campos de fecha:



Y en la última columna dispondremos de un par de Hyperlinks para actualizar o cancelar las modificaciones realizadas. Para finalizar, habilitaremos la paginación.

```
EditItemTemplate>
<asp:LinkButton id="LinkButton1" runat="server" __designer:wfdid="w49"
CommandName="Update"
ValidationGroup="editvalidation">Actualizar</asp:LinkButton>  <asp:LinkButton
id="LinkButton7" runat="server" __designer:wfdid="w50" CommandName="Cancel"
CausesValidation="False">Cancelar</asp:LinkButton>
</EditItemTemplate>
```



Una vez configurado el GridView de encuestas, añadiremos otro ObjectDataSource (odsRespuestas) que será el encargado de recoger la información referente a las respuestas relacionadas con una determinada encuesta.

Configuraremos dicho control del siguiente modo:

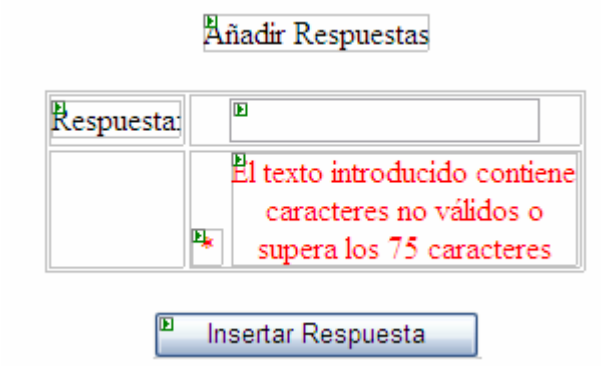
```
<asp:ObjectDataSource ID="odsRespuestas" runat="server"
   DataObjectTypeName="PortalWeb.BO.RespuestaEncuesta" DeleteMethod="Delete"
   OldValuesParameterFormatString="original_{0}" SelectMethod="ListByIdEncuesta"
    TypeName="PortalWeb.BLL.BLLRespuestaEncuesta" UpdateMethod="AddUpdate"
InsertMethod="AddUpdate">
      <SelectParameters>
         <asp:ControlParameter ControlID="gvEncuestas" Name="id"
PropertyName="SelectedValue"
            Type="Int32" />
      </SelectParameters>
   </asp:ObjectDataSource>
```

Lo siguiente será añadir un UpdatePanel y en su interior otro GridView, gvRespuestas. En este caso, el campo clave será _Id_Respuesta y el único campo a mostrar será la propia respuesta. Al igual que en el caso anterior, también habilitaremos la opción de paginación. Para posibilitar la opción de eliminar y editar respuestas, actuaremos exactamente de la misma forma en que lo hemos en el primero de los GridViews.

El panel de respuestas situado bajo el control Gridview de encuestas, sólo estará visible cuando se produzca el evento SelectIndexChanged, es decir, cuando seleccionemos alguna de las preguntas.

```
protected void gvEncuesta_SelectedIndexChanged(object sender, EventArgs e)
    {
        Panel2.Visible = true;
    }
```

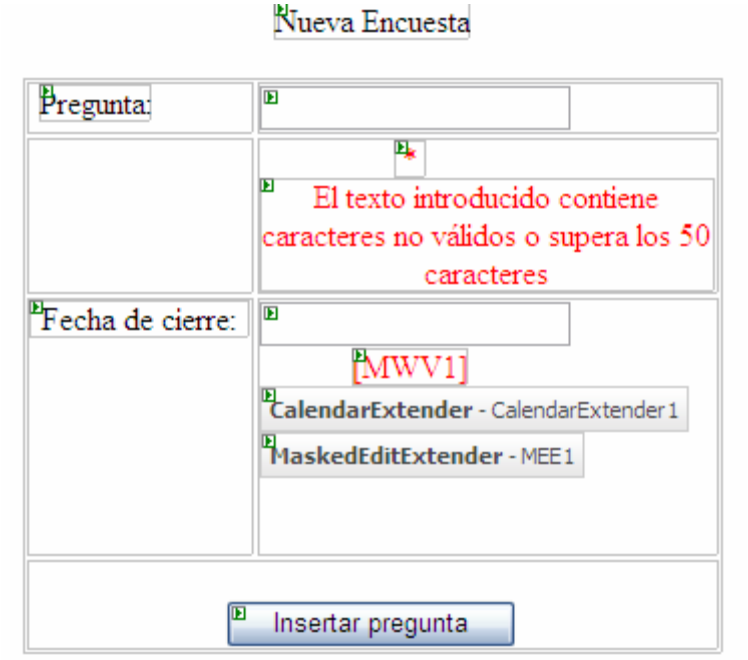
Con él se hará visible también una nueva tabla en la que podremos insertar las posibles respuestas a la pregunta seleccionada.



El método asociado al manejador del evento clic sobre el botón “Insertar Respuesta” es el siguiente:

```
protected void Añadir_Click(object sender, EventArgs e)
    {
        RespuestaEncuesta mirespuesta = new RespuestaEncuesta();
        mirespuesta._Id_Encuesta = int.Parse(gvEncuestas.SelectedValue.ToString());
        mirespuesta._Texto = ctRespuesta.Text;
        BLLRespuestaEncuesta.AddUpdate(mirespuesta);
        ctRespuesta.Text = "";
        gvRespuestas.DataBind();
    }
```

Para añadir una nueva encuesta crearemos un nuevo panel en la parte inferior de la página, en el que ubicaremos una tabla como la siguiente.



En ella recogeremos los datos correspondientes a la pregunta y a la fecha de cierre de la encuesta. Para ello haremos uso de controles TextBox, y sus correspondientes etiquetas indicando cada campo. También recurriremos, como hicimos en la sección de eventos, agregue los extenders CalendarExtender y MaskedEditExtender para recoger el valor de la fecha de cierre.

En cuanto al manejador del evento clic del botón “Insertar pregunta”, su código es el siguiente:

```
protected void Insertar_Click(object sender, EventArgs e)
    {
        Encuesta miencuesta = new Encuesta();
        miencuesta._Pregunta = ctPregunta.Text;
        miencuesta._Fecha_Publicación = DateTime.Now;
        miencuesta._Fecha_Cierre= Convert.ToDateTime(ctFechaCierre.Text);
        miencuesta._Activa= true;
        BLLEncuesta.AddUpdate(miencuesta);
        ctPregunta.Text = "";
        ctFechaCierre.Text = "";
        gvEncuestas.DataBind();
    }
```

Por último edite el controlador del evento Row_Updating del GridView relacionado con las encuestas:

```
protected void gvEncuesta_RowUpdating(object sender, GridViewUpdateEventArgs e)
    {
        DateTime dt;
        if (DateTime.TryParse(e.NewValues["_Fecha_Publicación"].ToString(), out dt))
        {
            e.NewValues["_Fecha_Publicación"] = dt;
        };
        if (DateTime.TryParse(e.NewValues["_Fecha_Cierre"].ToString(), out dt))
        {
            e.NewValues["_Fecha_Cierre"] = dt;
        };
    }
```

6.17.5. Gestión de Fuentes RSS

6.17.5.1. Descripción

Para la selección de aquellos canales o feeds de los que sustraeremos noticias para mostrar en nuestra página, recurriremos al modo de gestión de Fuentes RSS. Desde aquí podremos agregar nuevos canales, así como editar los ya existentes. Cada uno de ellos vendrá determinado principalmente por un título y una URL además de una pequeña descripción del contenido del mismo. También podremos decidir si queremos mantenerlo activo, es decir, mostrar sus noticias, o no.

6.17.5.2. Página GestionFuentesRSS.aspx

La página de gestión de feeds se dividirá en dos partes fundamentales: una para mostrar, editar y eliminar los canales de noticias RSS ya existentes; y la segunda para agregar nuevos canales.

Para la primera de ellas recurriremos al uso de un control Gridview, ubicado sobre un UpdatePanel, y enlazado debidamente con el correspondiente ObjectDataSource. Éste irá

conectado con el método SelectList(), de la clase BLLCanal de la capa de lógica de negocio, y que nos devolverá un listado de canales, es decir un objeto ListCanalRSS.

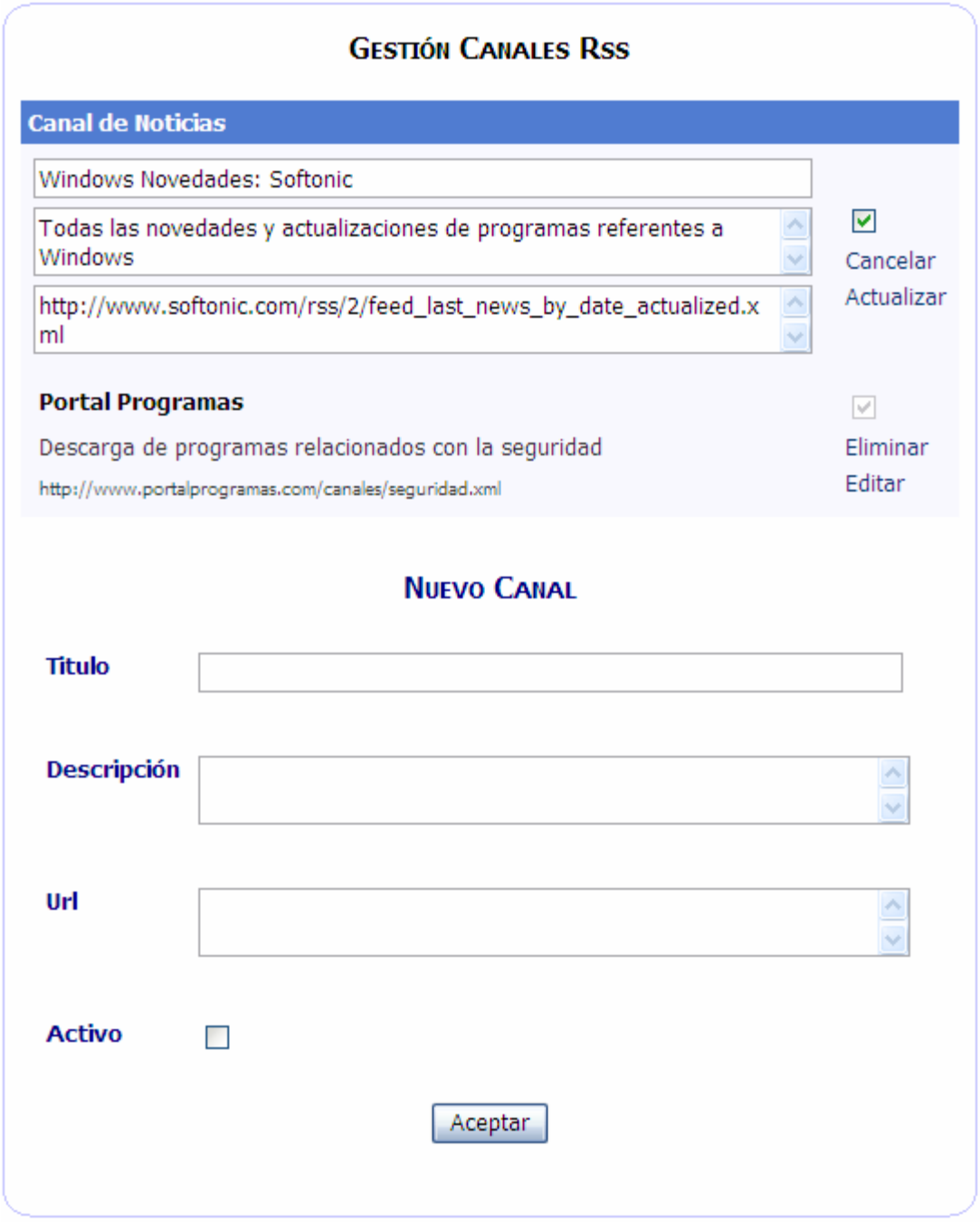
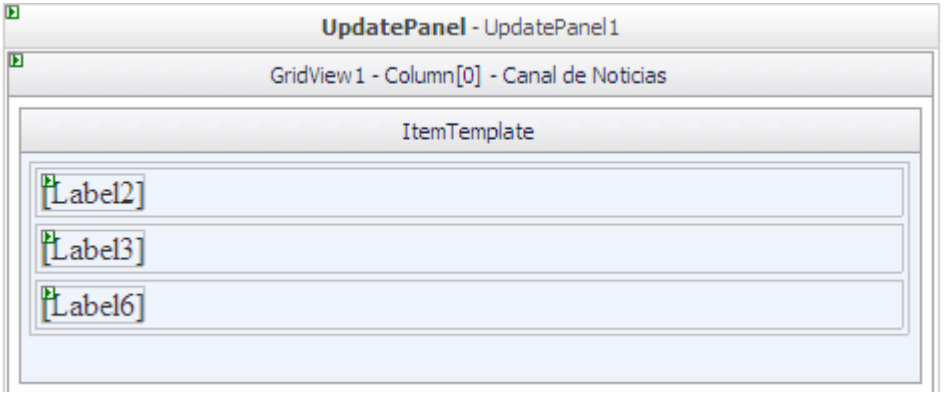


Fig. 104 Gestión de Canales RSS.

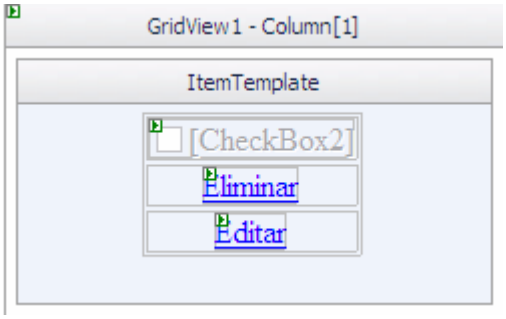
Cada uno de los elementos que conforman dicho GridView estará compuesto por un par de columnas. En la primera, insertaremos una tabla formada por tres filas, en las que irán ubicadas una serie de etiquetas enlazadas a determinados campos del objeto CanalRSS.

A continuación podemos observar la plantilla y su código correspondiente.




```
<ItemTemplate>
<TABLE style="OVERFLOW: auto" cellSpacing=3 cellPadding=3
width=440><TBODY><TR><TD align=left><asp:Label id="Label2" runat="server"
Text='<%# Eval("_Título") %>' CssClass="textosnegrita"
__designer:wfdid="w26"></asp:Label></TD></TR><TR><TD align=left><asp:Label
id="Label3" runat="server" Text='<%# Eval("_Descripción") %>'
__designer:wfdid="w27"></asp:Label></TD></TR><TR><TD align=left><asp:Label
id="Label6" runat="server" Text='<%# Eval("_Url") %>' CssClass="textospeq"
__designer:wfdid="w28"></asp:Label></TD></TR></TBODY></TABLE>
</ItemTemplate>
```

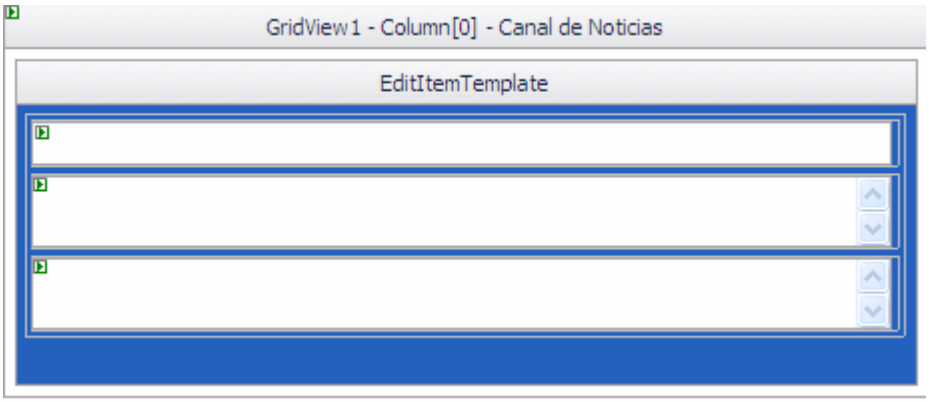
Para la segunda columna, destinada a albergar las distintas opciones que podremos ejecutar, emplearemos una plantilla como la siguiente:



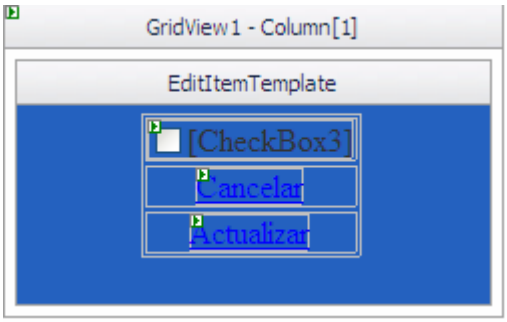

```
<ItemTemplate>
<TABLE><TBODY><TR><TD><asp:CheckBox id="CheckBox2" runat="server"
__designer:wfdid="w44" Checked='<%# Bind("_Activo") %>'
Enabled="False"></asp:CheckBox></TD></TR><TR><TD><asp:LinkButton id="LinkButton1"
runat="server" CssClass="textos" __designer:wfdid="w45" OnClientClick="return confirm
('¿Estas seguro de querer borrar este Canal?');"
```

```
CommandName="Delete">Eliminar</asp:LinkButton></TD></TR><TR><TD><asp:LinkButton
id="LinkButton2" runat="server" CssClass="textos" __designer:wfdid="w46"
CommandName="Edit">Editar</asp:LinkButton></TD></TR></TBODY></TABLE>
</ItemTemplate>
```

Ahora bien, para editar cada elemento recurriremos a la plantilla EditItemTemplate. La primera columna, presentará el siguiente aspecto: tres cajas de texto ubicadas en el interior de una tabla, destinadas a editar cada uno de los campos.

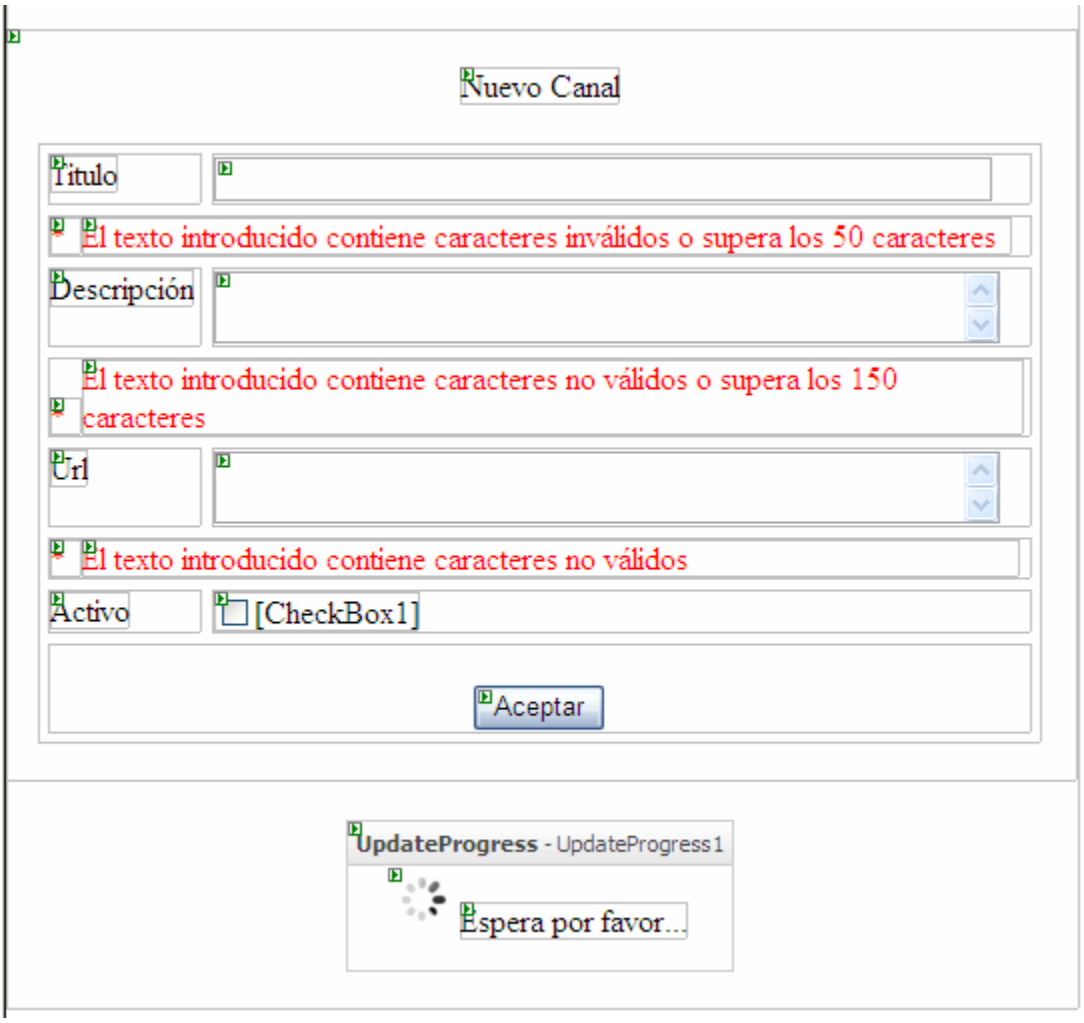


Y por otro lado, y ya en la segunda columna, contaremos con un Checkbox y con un par de enlaces para actualizar las modificaciones que hayamos realizado, o para, en caso contrario, cancelarlas.




```
<EditItemTemplate>
<TABLE><TBODY><TR><TD><asp:CheckBox id="CheckBox3" runat="server"
__designer:wfdid="w59" Checked='<%# Bind("_Activo")
%>'></asp:CheckBox></TD></TR><TR><TD><asp:LinkButton id="LinkButton1"
runat="server" CssClass="textos" __designer:wfdid="w60"
CommandName="Cancel">Cancelar</asp:LinkButton></TD></TR><TR><TD><asp:LinkButton
id="LinkButton2" runat="server" CssClass="textos" __designer:wfdid="w61"
CommandName="Update">Actualizar</asp:LinkButton></TD></TR></TBODY></TABLE>
</EditItemTemplate>
```

La segunda parte de la página de gestión de canales RSS se fundamentará en una tabla cuyos elementos de dispondrán de la siguiente forma.



Como se puede apreciar, en la tabla existirá una caja de texto para cada uno de los campos que conforman el objeto CanalRSS (excepto el identificador), así como un control CheckBox para indicar si cada elemento se encuentra activo o no.

Cada uno de estos campos estará acompañado por su etiqueta correspondiente, indicando su nombre, así como por una serie de controles del tipo RequiredFieldValidator para informarnos de su obligatoriedad o de ciertas restricciones a la hora de introducirlos.

Por último, se cuenta con un botón de aceptación para salvar todos los datos anteriormente introducidos, cuyo manejador del evento clic corresponde con el método añadir_canal.

Cada vez que añadamos un nuevo canal, crearemos pues un objeto CanalRSS cuyos campos se rellenarán con los datos anteriores, y que guardaremos de forma inmediata, tal y como muestra el siguiente fragmento de código:

```
protected void añadir_canal(object sender, EventArgs e)
    {
        CanalRSS micanal = new CanalRSS();
        micanal._Título = txtitulo.Text;
        micanal._Descripción = txtdescripcion.Text;
        micanal._Url = txturl.Text;
        micanal._Activo = CheckBox1.Checked;
        BLLCanalRSS.AddUpdate(micanal);
        txtitulo.Text = String.Empty;
        txtdescripcion.Text = String.Empty;
        txturl.Text = String.Empty;
        CheckBox1.Checked = false;
        GridView1.DataBind();
    }
```

Finalmente, mencionar que se cuenta con un control UpdateProgress, como en ocasiones anteriores, para que el usuario perciba de forma gráfica las evoluciones en la página.

6.17.6. Gestión de Usuarios

6.17.6.1. Descripción

Una de las tareas más importantes a la hora de mantener un sitio Web es la de gestionar los usuarios registrados en el mismo y administrar su información asociada. Para ello hemos diseñado un gestor de usuarios, cuyo interfaz es el siguiente.

Como se puede observar en la **Fig. 105**, en primer lugar se dispone de un listado de todos los usuarios registrados, en el que se muestran algunos aspectos claves de los mismos, como su dirección de correo electrónico, si están conectados en ese instante, o si se trata de usuarios activos.

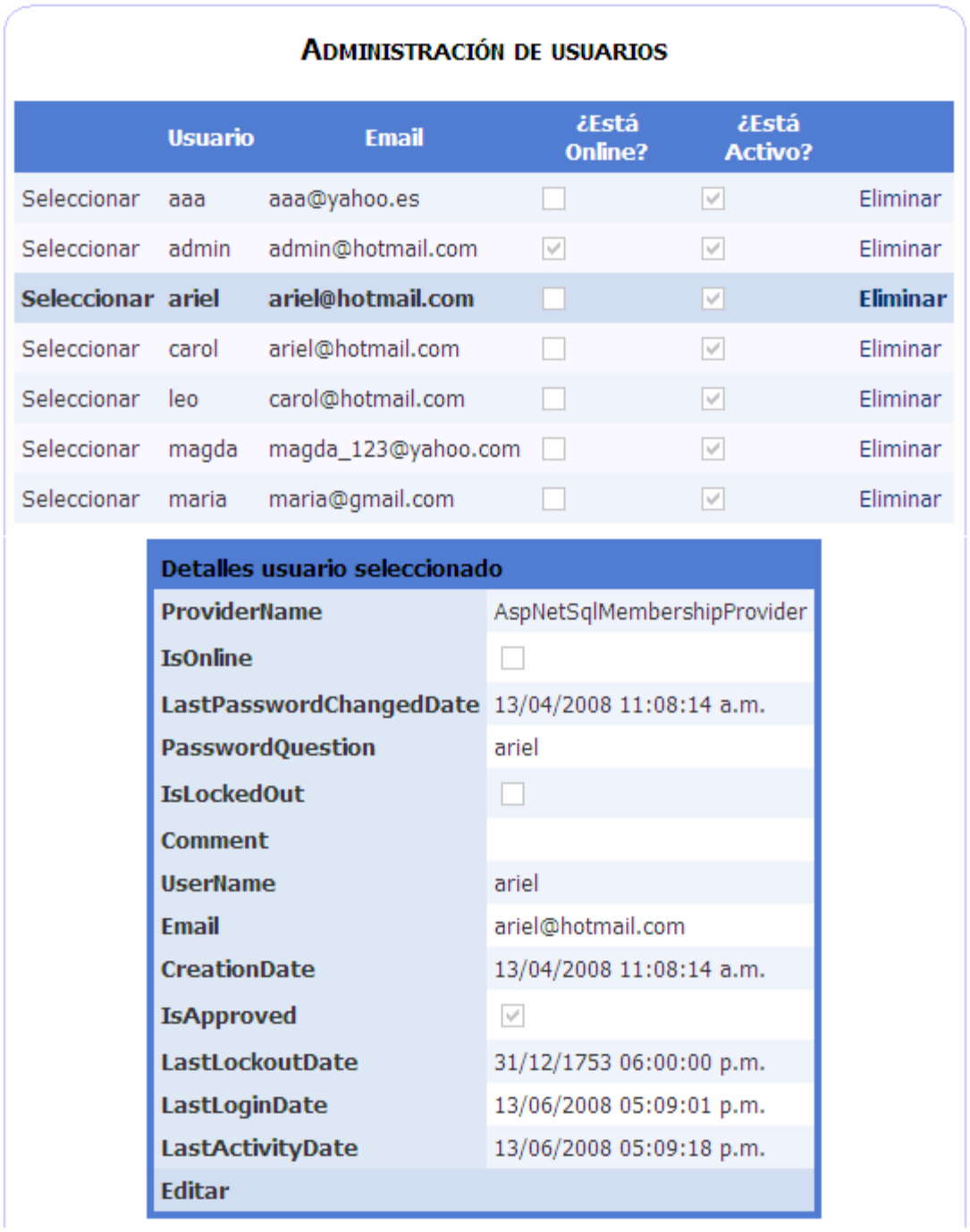


Fig. 105 Administración de Usuarios.

Si se desea conocer en detalle todos los aspectos relacionados con la cuenta de un usuario, se debe hacer clic en la opción seleccionar, y en la parte inferior se podrá visualizar toda la información relativa a dicho usuario. Datos como por ejemplo su pregunta secreta, o la última vez que inició sesión. Si se desea editar alguno de estos campos, debemos recurrir a la opción

editar que aparece en el final del listado. No todos los campos podrán ser manipulados. Si se desea eliminar un usuario, bastará con hacer clic en la opción eliminar.

Por otro lado, se tiene la posibilidad de gestionar la pertenencia a un rol de un usuario. Para ello hemos de asignarle algún rol, de entre los disponibles. Es desde este menú también, desde donde podremos dar lugar a nuevos roles, y también a eliminarlos.

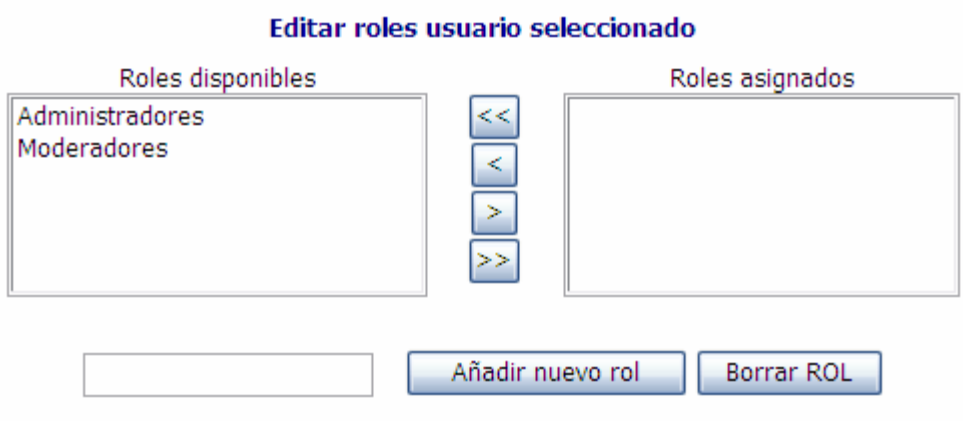


Fig. 106 Gestión de Roles.

Finalmente, cabe mencionar que también se puede crear y agregar nuevos usuarios desde ésta sección del modo administrador, con tan sólo introducir los datos correspondientes.

Fig. 107 Agregar un Nuevo Usuario.

6.17.6.2. Página Membership.aspx

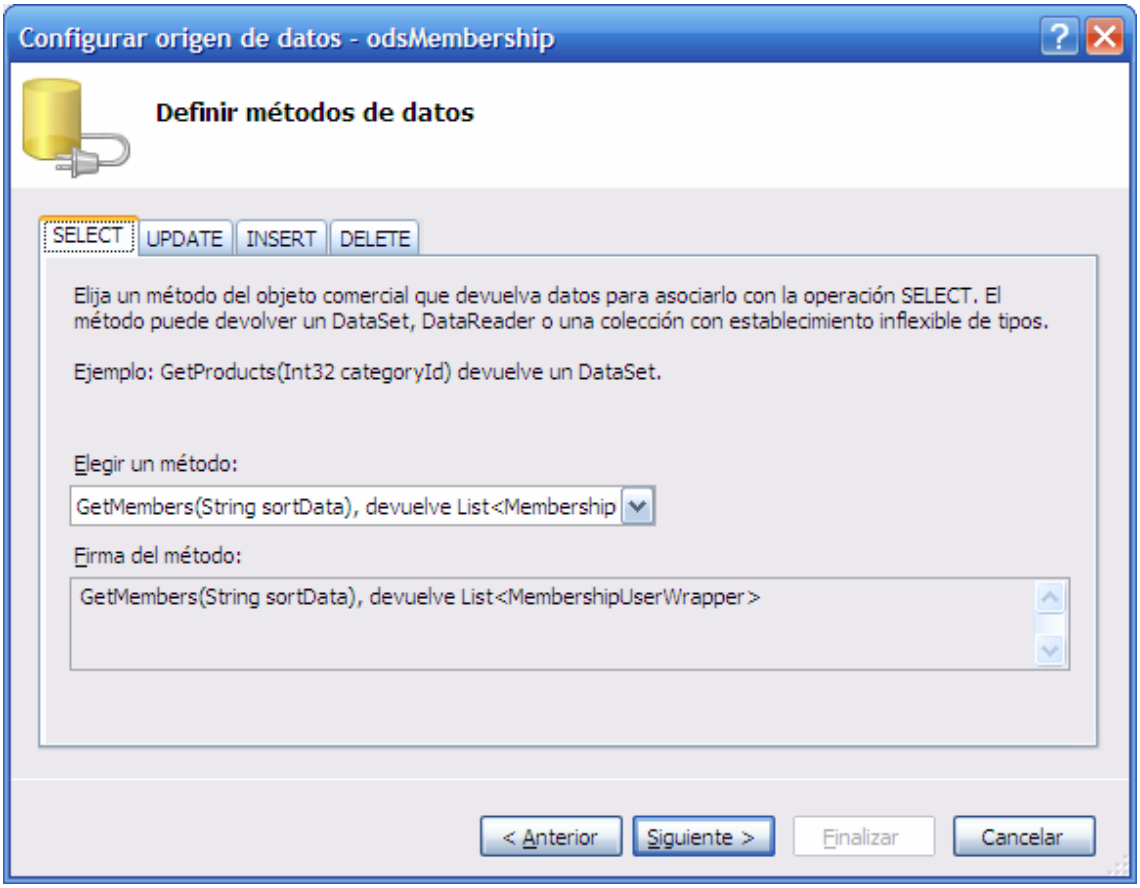
Para el desarrollo de ese control se hará uso de las clases de gestión de usuarios que insertamos anteriormente en nuestra carpeta App_Code/Business (*Ver Capítulo 1; Sección Administración de Miembros y Funciones*), así como directamente de los métodos proporcionados por el API de administración de pertenencia y funciones que ofrece ASP.NET.

Como punto de partida se empleará la página Membership.aspx que viene adjunta en el desarrollo de Peter Kellner que se ha instalado. A continuación ejecute los siguientes pasos:

Primeramente cree la página Membership.aspx, derivada de la principal (BasePage), y añada un panel junto con un extender RoundedCorners. Acto seguido inserte un control ObjectDataSource. Configure su origen de datos eligiendo para ello MembershipUserODS y elija como métodos asociados a las operaciones Select, Update, Insert y Delete las funciones de miembro de MembershipUserODS. En muchos casos, para cada una de estas funciones habrá varios métodos disponibles en la clase. De forma predeterminada, en las fichas se rellenarán los miembros marcados con el atributo especial [DataObjectMethod(DataObjectMethodType.Select, true)], en caso del Select. Por tanto en cada caso escoja el método seleccionado por defecto.

Además especifique como valor de la propiedad OldValuesParametersForms “{0}”. Una vez configurado el objeto ObjectDataSource, debe crear la interfaz del usuario. Para ello agregue un control GridView y lo asócielo al origen ObjectDataSource recién creado.

Dado que el Select devuelve muchos campos, al asociar el GridView es imposible sacar todos por pantalla. Para resolver este problema nos tenemos que ir a la vista de Código y establecer la propiedad Visible a false de aquellos campos que no nos van a interesar. En este caso el código resultaría de la siguiente forma:




```
<Columns>
<asp:CommandField ShowSelectButton="True"></asp:CommandField>
<asp:BoundField ReadOnly="True" DataField="UserName" SortExpression="UserName"
HeaderText="Usuario"></asp:BoundField>
<asp:BoundField DataField="Email" SortExpression="Email"
HeaderText="Email"></asp:BoundField>
<asp:CheckBoxField ReadOnly="True" DataField="IsOnline" SortExpression="IsOnline"
HeaderText="&#191;Est&#225; Online?"></asp:CheckBoxField>
<asp:CheckBoxField ReadOnly="True" DataField="IsLockedOut" Visible="False"
SortExpression="IsLockedOut" HeaderText="IsLockedOut"></asp:CheckBoxField>
<asp:CheckBoxField DataField="IsApproved" SortExpression="IsApproved"
HeaderText="&#191;Est&#225; Activo?"></asp:CheckBoxField>
<asp:BoundField ReadOnly="True" DataField="ProviderName" Visible="False"
SortExpression="ProviderName" HeaderText="ProviderName"></asp:BoundField>
<asp:BoundField ReadOnly="True" DataField="LastPasswordChangedDate"
Visible="False" SortExpression="LastPasswordChangedDate"
HeaderText="LastPasswordChangedDate"></asp:BoundField>
<asp:BoundField ReadOnly="True" DataField="PasswordQuestion" Visible="False"
SortExpression="PasswordQuestion"
HeaderText="PasswordQuestion"></asp:BoundField>
<asp:BoundField DataField="Comment" Visible="False" SortExpression="Comment"
HeaderText="Comment"></asp:BoundField>
<asp:BoundField ReadOnly="True" DataField="CreationDate" Visible="False"
SortExpression="CreationDate" HeaderText="CreationDate"></asp:BoundField>
```

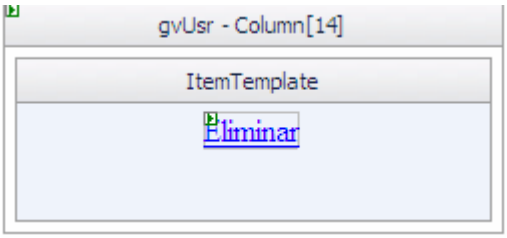
```
<asp:BoundField ReadOnly="True" DataField="LastLockoutDate" Visible="False"
SortExpression="LastLockoutDate" HeaderText="LastLockoutDate"></asp:BoundField>
<asp:BoundField DataField="LastLoginDate" Visible="False"
SortExpression="LastLoginDate" HeaderText="LastLoginDate"></asp:BoundField>
<asp:BoundField DataField="LastActivityDate" Visible="False"
SortExpression="LastActivityDate" HeaderText="LastActivityDate"></asp:BoundField>
<asp:TemplateField ShowHeader="False"><ItemTemplate>
<asp:LinkButton id="LinkButton1" runat="server" __designer:wfdid="w12"
OnClientClick="return confirm('¿Estás seguro de lo que vas a hacer?')"
CommandName="Delete">Eliminar</asp:LinkButton>
</ItemTemplate>
</asp:TemplateField>
</Columns>
```

El elemento DataKeyNames del control GridView se establece automáticamente. Esto se debe a que la clave principal se ha etiquetado en la clase MembershipUserWrapper con el atributo [DataObjectField(true)], como se muestra a continuación. Podemos observar igualmente, que debido a que UserName es una propiedad de la clase MembershipUser, fue preciso proporcionar una propiedad predeterminada en la clase que amplía MembershipUser. Dado que se trata de una propiedad de sólo lectura, sólo se declara un método Get (UserName es virtual y público en MembershipUser)

Ahora lo que se desea es poder seleccionar o eliminar los usuarios de la lista que va a mostrar el GridView. Para ello, en un primer lugar habilitaremos la opción de selección. También habilitaremos la opción paginación.

```
[DataObjectField(true)]
    public override string UserName
    {
        get { return base.UserName; }
    }
```

Para poder eliminar un usuario podríamos habilitar la eliminación de la misma manera, pero lo vamos a hacer de una manera alternativa. En el menú Editar Columnas, vamos a añadir un nuevo campo del tipo TemplateField. Seguidamente mediante la opción Editar Plantillas, editaremos la nueva columna.



Configúrelo de la siguiente manera:

```
<ItemTemplate>
<asp:LinkButton id="LinkButton1" runat="server" __designer:wfdid="w12"
OnClientClick="return confirm('¿Estás seguro de lo que vas a hacer?')"
CommandName="Delete">Eliminar</asp:LinkButton>
</ItemTemplate>
```

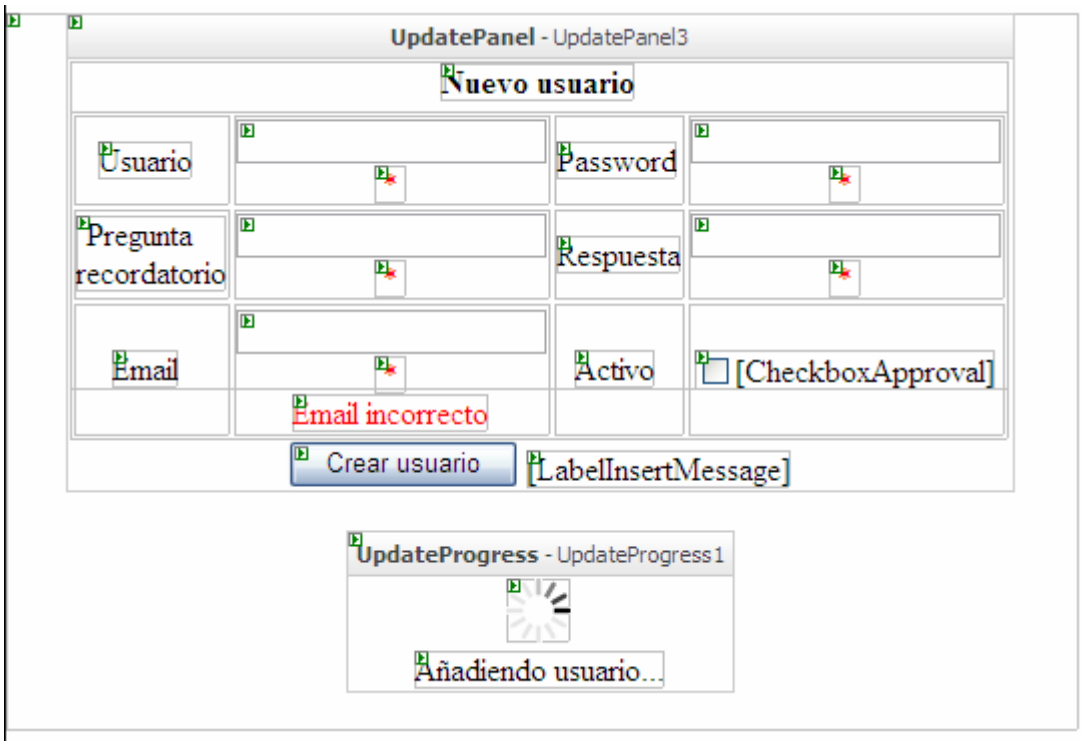
Lo siguiente será añadir un DataListView para ver los detalles de cada usuario. Este control irá cambiando en función del usuario seleccionado en el GridView. Una vez agregado el DataListView, lo asociaremos a un nuevo ObjectDataSource. Esta vez el método para Select no será el que está por defecto, si no este otro:

GetMembers(Boolean returnAllApprovedUsers, Boolean returnAllNotApprovedUsers, String usernameToFind, String sortData), retorna List<MembershipUserWrapper>

Deberemos de igual forma que en el ObjectDataSource anterior especificar como valor de la propiedad OldValuesParametersForms “{0}”. En este caso nos interesa habilitar la edición y la eliminación.

A continuación realice la interfaz para poder añadir usuarios nuevos a la aplicación.

Crearemos un formulario compuesto por diversos campos que habremos de introducir: Nombre de usuario, Contraseña, Pregunta Recordatorio, Repuesta de la pregunta, Email y si el usuario está activo o no. Cada uno de ellos estará formado por un control TextBox, acompañado de su correspondiente etiqueta, y junto a elementos RequiredFieldValidators que nos informarán de su obligatoriedad, todo esto dentro de un nuevo Panel y éste a su vez dentro de otro UpdatePanel



Al evento clic del botón "Crear usuario" le asociaremos un manejador llamado ButtonNewUser_Click y el código que ejecutará será el siguiente:

```
protected void ButtonNewUser_Click(object sender, EventArgs e)
    {
        odsMembership.InsertParameters["UserName"].DefaultValue = ctUsr.Text; ;
        odsMembership.InsertParameters["password"].DefaultValue = ctPassword.Text;
        odsMembership.InsertParameters["passwordQuestion"].DefaultValue =
ctPregunta.Text;
        odsMembership.InsertParameters["passwordAnswer"].DefaultValue =
ctRespuesta.Text;
        odsMembership.InsertParameters["email"].DefaultValue = ctEmail.Text;
        odsMembership.InsertParameters["isApproved"].DefaultValue =
CheckboxApproval.Checked == true ? "true" : "false";
        odsMembership.Insert();
        ctUsr.Text = "";
        ctPassword.Text = "";
        ctEmail.Text = "";
        ctPregunta.Text = "";
        ctRespuesta.Text = "";
        CheckboxApproval.Checked = false;
     }
```

Como se observa en el código, lo que hace este método es ejecutar el método Insert del objeto ObjectDataSource ObjectDataSourceMemebershipUser, pasándole como parámetros los valores introducidos en el formulario. A continuación vuelve a cargar los datos en el GridView para que aparezca el nuevo usuario, y restablece el formulario borrando los valores de las cajas de textos.

Lo siguiente que debe hacer es manejar el evento Inserted del objeto ObjectDataSource para saber si el usuario se ha insertado o no correctamente. El manejador de dicho evento tendrá el siguiente código:

```
protected void odsMembership_Inserted(object sender, ObjectDataSourceStatusEventArgs
e)
    {
        if (e.Exception != null)
        {
            LabelInsertMessage.Text = e.Exception.InnerException.Message + " No se ha
podido insertar al usuario";
            LabelInsertMessage.ForeColor = System.Drawing.Color.Red;
            e.ExceptionHandled = true;
        }
        else
        {
            LabelInsertMessage.Text = "Usuario " + ctUsr.Text + " insertado correctamente.";
            LabelInsertMessage.ForeColor = System.Drawing.Color.Green;
        }
    }
```

ROLES

Una vez que podemos seleccionar, editar, insertar y eliminar usuarios, nos interesa poder administrar los roles o funciones de nuestro sitio Web. Para ello, en primer lugar, agregaremos un nuevo objeto ObjectDataSource al que vamos a llamar odsRoles, y lo enlazaremos al objeto comercial RoleDataObject.ds. Como método para Select elegiremos: GetRoles(String userName, Boolean showOnlyAssignedRolls). Y configuraremos los parámetros de la siguiente manera:

```
<asp:ObjectDataSource id="odsRoles" runat="server"
__designer:dtid="281474976710694" TypeName="MembershipUtilities.RoleDataObject"
SelectMethod="GetRoles" InsertMethod="Insert" DeleteMethod="Delete"
__designer:wfdid="w30"
OldValuesParameterFormatString="original_{0}"><DeleteParameters>
```

```
<asp:Parameter Type="String" Name="roleName"></asp:Parameter>
</DeleteParameters>
<SelectParameters>
<asp:ControlParameter PropertyName="SelectedValue" Type="String" Name="userName"
ControlID="gvUsr"></asp:ControlParameter>
<asp:Parameter Type="Boolean" DefaultValue="false"
Name="showOnlyAssignedRolls"></asp:Parameter>
</SelectParameters>
<InsertParameters>
<asp:Parameter Type="String" Name="roleName"></asp:Parameter>
</InsertParameters>
</asp:ObjectDataSource>
```

Lo siguiente será definir una interfaz donde se puedan ver los roles disponibles y a su vez poderlos asociar o no a un determinado usuario. Principalmente, utilice dos ListBox, uno para roles disponibles y otro para roles asignados; cuatro botones para cambiar de un ListBox a otro (asociar todos los disponibles, asociar solo uno, desasociar todos los asignados, desasociar solo uno); una caja de texto y un botón para añadir nuevos roles, y un botón para eliminar un rol (Ver **Fig. 106)**

Para que cuando se seleccione un usuario en el GridView y se vea su información respecto a roles siga estas pautas:

Primeramente asocie un manejador al evento SelectedIndexChanged del GridView.

```
protected void gvUsr_SelectedIndexChanged(object sender, EventArgs e)
    {
        LabelInsertMessage.Text = "";
        GridView gv = (GridView)sender;
       RefrescarLbRoles(gv.SelectedValue.ToString());
    }
```

Este evento llama a una función llamada RefrescarLbRoles pasándole el nombre del usuario seleccionado. Su código será el siguiente:

```
private void RefrescarLbRoles(string usuario)
    {
        lbRoles.Items.Clear();
        lbRolesAsignados.Items.Clear();
        List<RoleData> lrd = RoleDataObject.GetRoles(usuario, false);
        foreach (RoleData rd in lrd)
        {
            if (rd.UserInRole)
                lbRolesAsignados.Items.Add(rd.RoleName);
            else
                lbRoles.Items.Add(rd.RoleName);
        }
```

```
        lblRolError.Text = "";
        LabelInsertMessage.Text = "";
    }
```

Esta función borrará los elementos de los dos ListBox, consultará los roles disponibles en nuestra aplicación y según estén o no asignados al usuario en cuestión los añadirá a un ListBox o a otro.

Cuando se cargue por primera vez la página, el evento SelectIndexChanged del GridView no se habrá producido. Sin embargo, por defecto, el primer usuario del mismo estará seleccionado. Para que los ListBox se rellenen en función de los roles asignados o no a este usuario tendremos que crear otro método llamado FindFirstUserName(), y le llamaremos cuando carguemos la página por primera vez.

```
private void FindFirstUserName()
    {
        if (gvUsr.Rows.Count > 0)
            gvUsr.SelectedIndex = 0;
        string username;
        username = gvUsr.SelectedValue.ToString();
        RefrescarLbRoles(username);
    }
```

Este método pasa a la función RefrescarLbRoles, el nombre del primer usuario seleccionado. Nos quedaría tan sólo configurar los eventos clic de los diferentes botones que hemos colocado en el formulario. Para crear un nuevo rol usaremos este código:

```
protected void ButtonCreateNewRole_Click(object sender, EventArgs e)
    {
        if (TextBoxCreateNewRole.Text.Length > 0)
        {
            odsRoles.InsertParameters["RoleName"].DefaultValue =
TextBoxCreateNewRole.Text; ;
            odsRoles.Insert();
            RefrescarLbRoles(gvUsr.SelectedValue.ToString());
            TextBoxCreateNewRole.Text = "";
        }
    }
```

Y para borrar un rol:

```
protected void btnBorrarRol_Click(object sender, EventArgs e)
   {
      if (lbRoles.SelectedIndex >= 0)
      {
         try
         {
           RoleDataObject.Delete(lbRoles.SelectedValue.ToString());
            RefrescarLbRoles(gvUsr.SelectedValue.ToString());
         }
         catch
         {
            lblRolError.Text = "No se puede borrar un rol si hay usuarios que lo tienen
asignado.";
            lblRolError.ForeColor = System.Drawing.Color.Red;
         }
      }
      else
      {
         lblRolError.Text = "No ha seleccionado un rol dentro de los disponibles";
         lblRolError.ForeColor = System.Drawing.Color.Red;
      }
   }
```

Por último, el código para asociar y desasociar roles a un usuario será el siguiente:

```
protected void btnDesAsignaAll_Click(object sender, EventArgs e)
   {
      if (lbRolesAsignados.Items.Count > 0)
      {
         string[] usuarios = new string[1];
         for (int i = 0; i < lbRolesAsignados.Items.Count; i++)
         {
            usuarios[0] = gvUsr.SelectedValue.ToString();
            Roles.RemoveUsersFromRole(usuarios,
               lbRolesAsignados.Items[i].ToString());
         }
         RefrescarLbRoles(gvUsr.SelectedValue.ToString());
      }
   }

   protected void btnDesasignaRol_Click(object sender, EventArgs e)
   {
      if (lbRolesAsignados.SelectedIndex >= 0)
      {
         string[] usuarios = new string[1];
         usuarios[0] = gvUsr.SelectedValue.ToString();
         Roles.RemoveUsersFromRole(usuarios,
            lbRolesAsignados.SelectedValue.ToString());
         RefrescarLbRoles(gvUsr.SelectedValue.ToString());
      }
   }
```

```
protected void btnAsignaRol_Click(object sender, EventArgs e)
{
    if (lbRoles.SelectedIndex >= 0)
    {
        Roles.AddUserToRole(gvUsr.SelectedValue.ToString(),
            lbRoles.SelectedValue.ToString());
        RefrescarLbRoles(gvUsr.SelectedValue.ToString());
    }
}
protected void btnAsignaAll_Click(object sender, EventArgs e)
{
    if (lbRoles.Items.Count > 0)
    {
        for (int i = 0; i < lbRoles.Items.Count; i++)
        {
            Roles.AddUserToRole(gvUsr.SelectedValue.ToString(),
                lbRoles.Items[i].ToString());
        }
        RefrescarLbRoles(gvUsr.SelectedValue.ToString());
    }
}
```

Para concluir, será necesario añadir dos controles UpdatePanel, en cuyo interior se ubicaran los diferentes controles anteriormente descritos. El primer UpdatePanel albergará el GridView y el DataList, y tendrá asociado un UpdateProgress. Por otro lado, el segundo UpdatePanel contendrá el formulario de creación de nuevo usuario. Una vez, agregados dichos paneles, añadiremos un manejador para el evento PageIndexChanging (gvUsr_PageIndexChanging), y otro para el evento Row_Deleted (gvUsr_RowDeleted), cuyas respectivas definiciones son las siguientes:

```
protected void gvUsr_RowDeleted(object sender, GridViewDeletedEventArgs e)
    {
        gvUsr.SelectedIndex = 0;
        string username;
        username = gvUsr.SelectedValue.ToString();
       RefrescarLbRoles(username);
    }

    protected void gvUsr_PageIndexChanging(object sender, GridViewPageEventArgs e)
    {
        gvUsr.SelectedIndex = 0;
    }
```

6.17.7. Gestión de Foros

6.17.7.1. Descripción

Para completar las tareas de moderación del foro y poder administrar en profundidad los temas y contenidos de la sección de foros, se ha creado el modo de gestión de foros. Desde esta perspectiva se podrán editar y eliminar los foros ya existentes, así como dar lugar a foros nuevos.

Fig. 108 Gestión de Foros.

Para visualizar los temas contenidos en alguno de los foros ya existentes, bastará con hacer clic sobre cualquiera de ellos. Para que se despliegue el listado de mensajes de cada uno de dichos temas, tan sólo tendrá que seleccionar aquel que desee.

Tanto para los mensajes como para los temas se tendrán habilitadas una serie de opciones. Por un lado podrá desactivar o reactivar un determinado tema, así como eliminarlo definitivamente. Y por otro lado podrá ocultar o dejar visible un mensaje.

Para la primera de las opciones, se tendrá que hacer clic en el botón superior que aparece junto a cada mensaje. Y para la segunda, en el situado justo debajo.

Fig. 109 Activar o cerrar un Foro.

6.17.7.2. Página GestiónForo.aspx

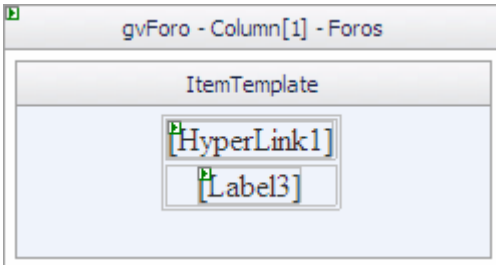
La primera de las páginas, dedicada a gestionar de forma global el conjunto de foros, contará con dos partes diferenciadas: la parte superior, destinada a posibilitar la edición de los foros existentes; y la inferior, diseñada para la creación de nuevos foros.

Para la primera parte se contará con un control GridView (gvForo) en el que se muestran diversos campos relevantes relativos a la tabla Foros. Enlácelo con un ObjectDataSource

```
<asp:ObjectDataSource ID="odsForo" runat="server"
DataObjectTypeName="PortalWeb.BO.Foro" DeleteMethod="Delete"
EnablePaging="True" OldValuesParameterFormatString="original_{0}"
SelectCountMethod="Foro_Num" SelectMethod="Foro_ListPaginados"
TypeName="PortalWeb.BLL.BLLForo" UpdateMethod="AddUpdate"
MaximumRowsParameterName="pageSize" StartRowIndexParameterName="pageIndex">
 </asp:ObjectDataSource>
```

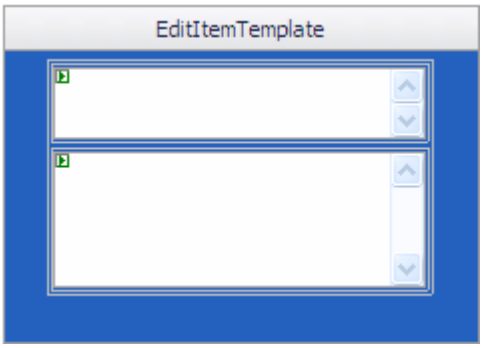
Los campos a mostrar abarcarán, desde el nombre del foro, hasta los datos de su creación, pasando por el número de mensajes y temas contenidos en cada uno de ellos.

En el GridView habilite la paginación como el modo edición. La única columna editable será la primera, es decir, la que contiene el nombre y la descripción del foro en cuestión. Por tanto, haremos uso de dos plantillas: la ItemTemplate para mostrar los elementos, y la EditItemTemplate para editarlos.



Como se puede observar, dentro de una pequeña tabla, está un primer campo, de tipo Hyperlynk enlazado con nombre del foro, que a posteriori nos servirá cono enlace con el conjunto de temas asociados a dicho foto (NavigateUrl = Eval("_Id_Foro", "~//Editores//GestiónMensajes.aspx?id={0}")) inmediatamente debajo de él, una etiqueta enlazada con el correspondiente campo _Descripción.

Cuando pase a la plantilla de edición, su aspecto será el siguiente:

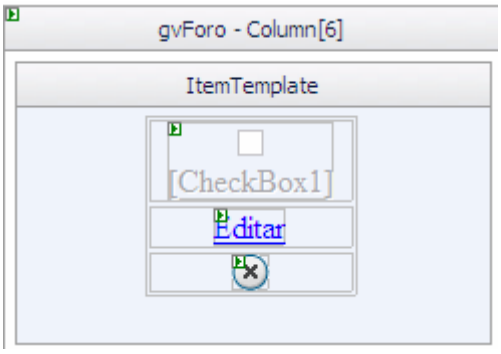


Y éste será el código que lo acompañe:

```
<EditItemTemplate>
   <table>
        <tr>
            <td>
                <asp:TextBox ID="TextBox1" runat="server" CssClass="textos" Font-
Bold="True" Rows="2" Text='<%# Bind("_Nombre") %>' TextMode="MultiLine"
Width="180px">
                </asp:TextBox></td>
           </tr>
            <tr>
              <td>
                <asp:TextBox ID="TextBox2" runat="server" CssClass="textos" Rows="4"
Text='<%# Bind("_Descripción") %>' TextMode="MultiLine"
Width="180px"></asp:TextBox></td>
            </tr>
    </table>
</EditItemTemplate>
```

El resto de columnas tan sólo tendrán etiquetas enlazadas a los correspondientes campos de datos en su plantilla ItemTemplate. Mención aparte merece la última de las columnas. En ella hemos insertado una pequeña tabla con tres controles. El primero de ellos es un CheckBox destinado a manipular la propiedad "_Activo" de cada foro. El segundo es el botón Editar con el que se nos mostrará la fila correspondiente en su modo edición. Y por último contaremos con otro botón, basado en un pequeño gif, destinado a eliminar directamente dicho registro de la base de datos.

Cuando estemos en el modo edición, el aspecto de la segunda fila cambiará, y en vez de encontrarnos con el botón "Editar", nos encontraremos con los botones "Actualizar" y "Cancelar" para operar sobre los datos que estamos manipulando.



Éste será el código asociado a la plantilla de edición de la última columna, en su modo edición.

```
<EditItemTemplate>
    <table>
            <tr>
                <td style="width: 100px">
                    <asp:CheckBox ID="CheckBox1" runat="server" Checked='<%#
Bind("_Activo") %>' /></td>
            </tr>
            <tr>
                <td style="width: 100px">
                    <asp:LinkButton ID="LinkButton1" runat="server"
CausesValidation="False" CommandName="Update">Actualizar</asp:LinkButton></td>
            </tr>
            <tr>
                <td style="width: 100px">
                    <asp:LinkButton ID="LinkButton2" runat="server"
CommandName="Cancel">Cancelar</asp:LinkButton></td>
            </tr>
            <tr>
                <td style="width: 100px">
                    <asp:ImageButton ID="ImageButton1" runat="server"
CommandName="Delete" ImageUrl="~/Controles/foro/imagenes/topic_delete.gif"
                        OnClientClick="return confirm ('¿Estas seguro de querer borrar
este foro?');" /></td>
            </tr>
        </table>
</EditItemTemplate>
```

Como se puede observar, las operaciones de cancelar y actualizar están asociadas a los métodos correspondientes. También se comprueba que antes de borrar un registro se pedirá confirmación por pantalla.

En lo que respecta a la parte inferior de la página, sitúe una tabla con dos columnas y múltiples filas, en las que se insertarán diversos controles. Se cuenta con controles TextBox para los campos nombre y descripción, así como los correspondientes RequiredFieldValidators para

informar de su obligatoriedad. Parejos a todos ellos, y en la columna izquierda, ubique las etiquetas asociadas a dichos campos. Finalmente, y para el campo “Activo”, emplearemos un control CheckBox.

Justo debajo de la tabla, agregue un botón Añadir, cuyo manejador estará asociado al método que se observa a continuación.

```
protected void añadir_foro(object sender, EventArgs e)
    {
        Foro miforo = new Foro();
        miforo._Nombre = ctNombreForo.Text;
        miforo._Descripción = ctDespForo.Text;
        miforo._Fecha_Creación = DateTime.Today;
        miforo._Creador = HttpContext.Current.User.Identity.Name.ToString();
        miforo._Activo = cvForoActivo.Checked;
        ctNombreForo.Text = "";
        ctDespForo.Text = "";
        cvForoActivo.Checked = true;
        BLLForo.AddUpdate(miforo);
        gvForo.DataBind();
    }
```

Otros métodos necesarios para el correcto funcionamiento de esta sección, son los que e muestran a continuación:

```
protected void gvForo_RowUpdating(object sender, GridViewUpdateEventArgs e)
    {
        DateTime dt;
        if (DateTime.TryParse(e.NewValues["_Fecha_Creación"].ToString(), out dt))
        {
            e.NewValues["_Fecha_Creación"] = dt;
        };
    }

protected string GetNumMens(object idforo)
    {
        return  BLLMensaje.NumById_Foro((int)idforo).ToString();
    }

protected string GetNumTemas(object idforo)
    {
        return BLLMensaje.NumPrincipalesById_Foro((int)idforo).ToString();
    }
```

Finalmente, cabe mencionar que los controles existentes en esta página serán albergados por un gran UpdatePanel, y éste dentro de un Panel, al que aplicaremos, como venimos haciendo hasta ahora, la técnica de esquinas redondeadas.

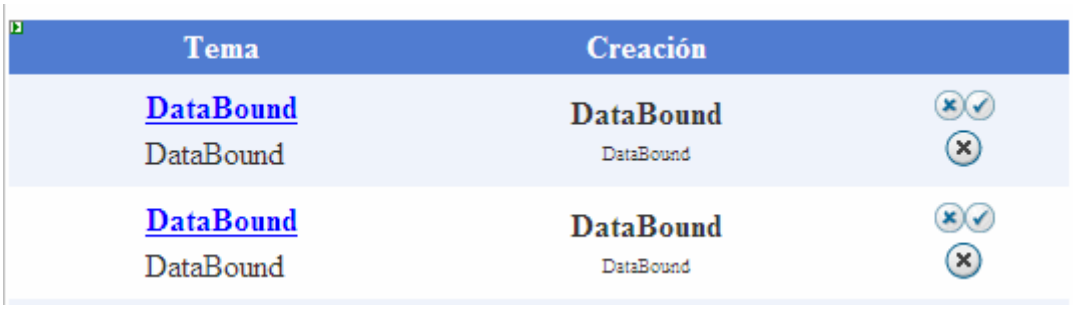
6.17.7.3. Página GestiónMensajes.aspx

Para la gestión de los mensajes pertenecientes a cada uno de los foros, se recurrirá principalmente, al uso de un par de controles GridView (gvMensajes, gvTemas).

El primero de ellos, enlácelo un ObjectDataSource tal y como se muestra en el siguiente código:

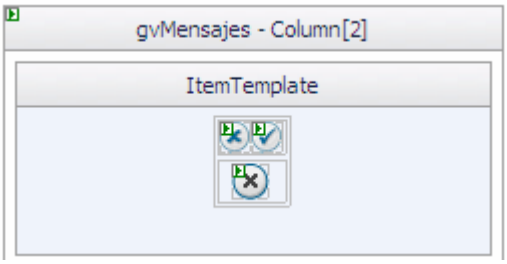
```
<asp:ObjectDataSource id="odsMensajesPrincipales" runat="server"
StartRowIndexParameterName="pageIndex" MaximumRowsParameterName="pageSize"
EnablePaging="True" SelectCountMethod="NumPrincipalesById_Foro"
UpdateMethod="AddUpdate" DataObjectTypeName="PortalWeb.BO.Mensaje"
OldValuesParameterFormatString="original_{0}"
SelectMethod="ListHuerfanosById_ForoPaginados" DeleteMethod="Delete"
TypeName="PortalWeb.BLL.BLLMensaje" __designer:wfdid="w120"><SelectParameters>
<asp:QueryStringParameter Type="Int32" Name="id"
QueryStringField="id"></asp:QueryStringParameter>
</SelectParameters>
</asp:ObjectDataSource>
```

En este GridView se nos mostrarán, de forma paginada, todos los temas que componen un determinado foro. Además, podremos observar su descripción, y algunos detalles de su creación.



Para representar todos estos campos, use etiquetas, enlazadas debidamente con cada campo correspondiente. Para este caso, sólo emplearemos plantillas de tipo ItemTemplate, puesto que no podremos editar ni los temas ni los mensajes existentes.

Al igual que en la página anterior, en la última columna se dispone de una serie de botones basados en gifs, cuya misión será la de eliminar el tema (botón inferior); o cerrarlo (botones superiores). Dependiendo de si el tema está actualmente abierto o no, se nos mostrará uno u otro en cada caso.



A continuación se muestra el código correspondiente a dichos botones:

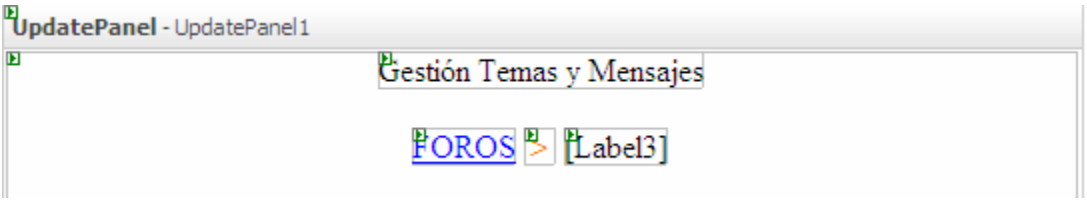
```
<ItemTemplate>
<TABLE __designer:dtid="2814749767106603"><TBODY><TR
__designer:dtid="2814749767106607"><TD
__designer:dtid="2814749767106608"><asp:ImageButton id="ImageButton2"
runat="server" __designer:wfdid="w167" ImageUrl="~/Controles/foro/imagenes/icon6.gif"
CommandName="Cerrar" OnClientClick="return confirm ('¿Estas seguro de querer cerrar
este mensaje?');" CausesValidation="False" CommandArgument='<%#
Eval("_Id_Mensaje") %>' OnCommand="cerrar_tema" Visible='<%#
GetActivo1(Eval("_Activo")) %>'></asp:ImageButton><asp:ImageButton
id="ImageButton3" runat="server" __designer:wfdid="w168"
ImageUrl="~/Controles/foro/imagenes/icon5.gif" CommandName="Reactivar"
OnClientClick="return confirm ('¿Estas seguro de querer activar este mensaje?');"
CausesValidation="False" CommandArgument='<%# Eval("_Id_Mensaje") %>'
OnCommand="activar_tema" Visible='<%# GetActivo2(Eval("_Activo"))
%>'></asp:ImageButton></TD></TR><TR __designer:dtid="2814749767106610"><TD
__designer:dtid="2814749767106611"><asp:ImageButton id="ImageButton1"
runat="server" __designer:dtid="2814749767106612" __designer:wfdid="w169"
ImageUrl="~/Controles/foro/imagenes/topic_delete.gif" CommandName="Delete"
OnClientClick="return confirm ('¿Estas seguro de querer borrar este foro?');"
CausesValidation="False"></asp:ImageButton></TD></TR></TBODY></TABLE>
</ItemTemplate>
```

Y también el correspondiente a los manejadores de los dos primeros: cerrar y activar tema:

```
protected void cerrar_tema(object sender, CommandEventArgs e)
    {
        Mensaje mimensaje;
        int id = Convert.ToInt32(e.CommandArgument.ToString());
        mimensaje = BLLMensaje.SelectGetById(id);
        mimensaje._Activo = false;
        BLLMensaje.AddUpdate(mimensaje);
        gvTemas.DataBind();
        gvMensajes.DataBind();
    }

protected void activar_tema(object sender, CommandEventArgs e)
    {
        Mensaje mimensaje;
```

```
        int id = Convert.ToInt32(e.CommandArgument.ToString());
        mimensaje = BLLMensaje.SelectGetById(id);
        mimensaje._Activo = true;
        BLLMensaje.AddUpdate(mimensaje);
        gvTemas.DataBind();
        gvMensajes.DataBind();
    }
```

Además, en la parte superior de la página, contaremos con una etiqueta para ver de donde venimos y mostrar el nombre del foro en el que nos encontramos, cuyo texto obtendremos de la siguiente forma:




```
protected void Page_Load(object sender, EventArgs e)
    {
      int idforo = Convert.ToInt32(Request.QueryString["id"].ToString());
       Foro miforo = new Foro();
       miforo = BLLForo.SelectGetById(idforo);
       nombre = miforo._Nombre;
       Label3.Text = nombre;
    }
```

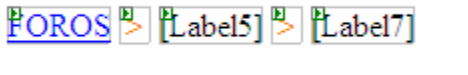
En cuanto a la segunda mitad de la página, el Gridview que se emplea, funciona de forma similar a la detallada anteriormente. Con la excepción de que los mensajes en lugar de activarlos o desactivarlos, los dejaremos visibles o los ocultaremos.

Su ObjectDataSource está configurado de la siguiente manera:

```
<asp:ObjectDataSource id="odsMensajes" runat="server"
StartRowIndexParameterName="pageIndex" MaximumRowsParameterName="pageSize"
EnablePaging="True" SelectCountMethod="NumById_Padre"
UpdateMethod="AddUpdate" DataObjectTypeName="PortalWeb.BO.Mensaje"
OldValuesParameterFormatString="original_{0}"
SelectMethod="ListById_PadrePaginados" DeleteMethod="Delete"
TypeName="PortalWeb.BLL.BLLMensaje" __designer:wfdid="w137"><SelectParameters>
<asp:ControlParameter PropertyName="SelectedValue" Type="Int32" Name="id"
```

```
ControlID="gvMensajes"></asp:ControlParameter>
</SelectParameters>
</asp:ObjectDataSource>
```

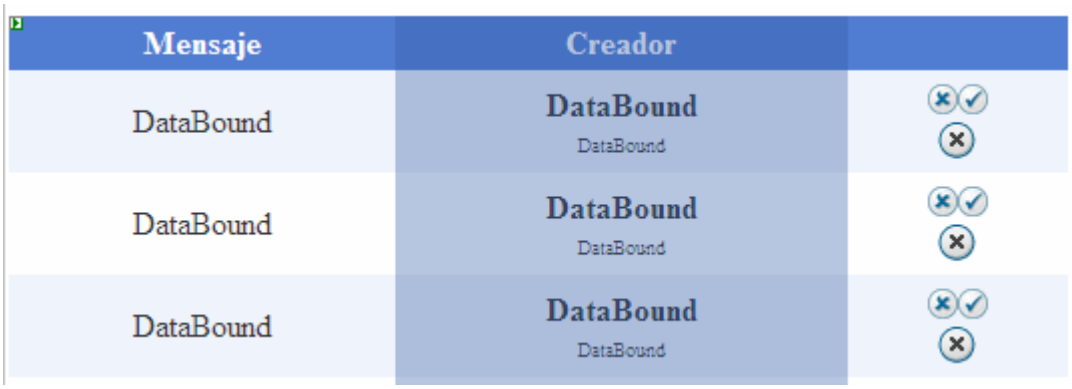
También contaremos con una serie de etiquetas para ubicarnos en el tema que hayamos seleccionado en el GridView superior, y que cambiará dinámicamente dependiendo de qué índice seleccionemos.



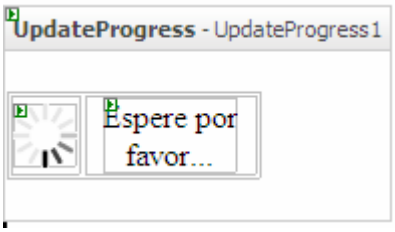
La primera de ellas, Label5, nos indicará el foro; y Label7 el tema. Todas ellas estarán o no visibles dependiendo de si hemos seleccionado o no algún tema en el GridView superior.

```
protected void gvMensajes_SelectedIndexChanged1(object sender, EventArgs e)
   {
       HyperLink2.Visible = Label4.Visible = Label5.Visible = Label6.Visible = Label7.Visible
= true;
       Label5.Text = nombre;
       gvTemas.PageIndex = 0;
       Mensaje mimensaje = new Mensaje();
       mimensaje = BLLMensaje.SelectGetById((int)this.gvMensajes.SelectedValue);
       Label7.Text = mimensaje._Título;
       gvTemas.DataBind();
  }
```

En cuanto a los campos del GridView, se mostrarán tan sólo el texto del mensaje, y algunos detalles acerca de su creación. Sobre cada uno de ellos podremos, eliminarlo directamente, o modificar su propiedad Visible. Para ello procederemos exactamente de la misma forma que en el caso anterior. En este caso ninguno de los campos constituirá un Hyperlink: todos serán controles de tipo Label.



Finalmente, comentar que como en otras ocasiones, incluiremos un control UpdateProgress al final de la página para hacer visibles los cambios en esta, y que estará asociado al correspondiente UpdatePanel.



Con esto termina la gestión del Portal sin, embargo, para que sólo los usuarios con rol de Administradores, accedan a todo el contenido de la carpeta Editores, añada dentro de esta un nuevo archivo de configuración, y agregue una regla, que permita el acceso a usuarios con rol de “Administradores”, y se deniegue el acceso al resto de usuarios.

```
<configuration xmlns="http://schemas.microsoft.com/.NetConfiguration/v2.0">
  <system.web>
    <authorization>
      <allow roles="Administradores"/>
      <deny users="*" />
    </authorization>
  </system.web>
</configuration>
```

6.18. Personalización del Sitio Web

6.18.1. Introducción

Para posibilitar la personalización del sitio Web por parte de los usuarios se ha apostado por el uso de temas mediante la propiedad que nos ofrece ASP.NET 2.0.

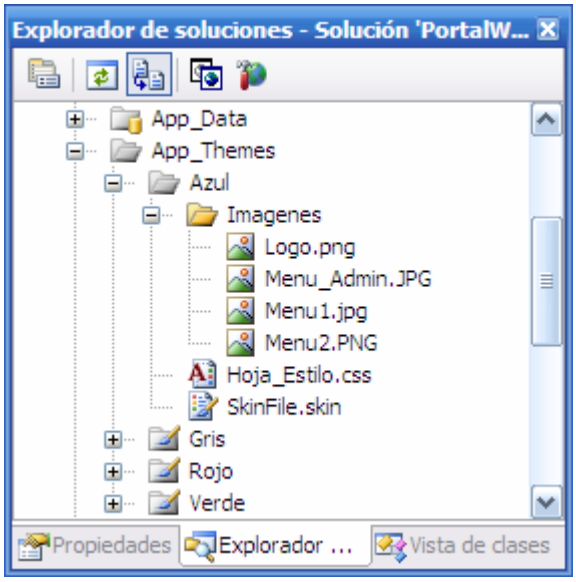
A la hora de elegir o cambiar alguno de esos temas, el usuario previamente autenticado deberá dirigirse al menú Login ubicado en la página maestra, y seleccionar un tema de entre los existentes en una lista desplegable. Cabe resaltar que se trata de una funcionalidad disponible sólo para los usuarios registrados.

Con tan sólo seleccionar uno de ellos, los cambios en el aspecto de la Web se aplicarán a todos y cada uno de los paneles que componen la página principal, así como también al menú y al logo situado en la cabecera.

El tema seleccionado por el usuario se mantendrá de tal forma que seguirá conservando la configuración que seleccionó, cada vez que vuelva a realizar una visita al portal en el futuro.

6.18.2. Desarrollo

Para implementar esta funcionalidad, se crearán varias carpetas en el directorio de ASP.NET App_Themes, una para cada tema. En su interior, cada una de ellas, tendrá a su vez una carpeta denominada Imágenes que contendrá todas aquellas imágenes disponibles para ese tema, además de una hoja de estilos .css y un archivo de máscaras .skin. (*Capítulo 1; Sección Temas y Máscaras de ASP .NET 2.0*)



En un futuro nos interesará determinar de forma exacta cuántos temas existen en la carpeta App_Themes. Para ello, crearemos un objeto de negocio llamado Tema, cuya única propiedad será una cadena de texto denominada _Nombre.

```
namespace PortalWeb.BO
{
public class Tema
{
      private string Nombre;

      public string _Nombre
      {
         get
         {
            return Nombre;
         }
         set
         {
            Nombre = value;
         }
      }
   }
}
```

Además cree la colección ListTema, de la misma forma que hicimos con los anteriores objetos de negocio.

Por otro lado, en la capa de lógica de negocio, añadiremos una nueva clase llamada BLLTema. Ésta estará formada por un solo método, GetTemas(), que devolverá un objeto del tipo TemaList, es decir una colección de objetos de tipo tema.

```
[DataObjectAttribute()]
    public class BLLTema
    {
        public ListTema GetTemas()
        {
            DirectoryInfo dInfo = new
DirectoryInfo(System.Web.HttpContext.Current.Server.MapPath("~/App_Themes"));
            DirectoryInfo[] dArrInfo = dInfo.GetDirectories();
            ListTema temas = new ListTema();
            foreach (DirectoryInfo sDirectory in dArrInfo)
            {
                Tema mitema = new Tema();
                mitema._Nombre = sDirectory.Name;
                temas.Add(mitema);
            }
            return temas;
        }
    }
```

Como apreciamos en el código, en primer lugar el método crea un objeto de tipo DirectoryInfo destinado a albergar la información referente al directorio AppThemes de nuestro proyecto. Seguidamente, y mediante el método GetDirectories, almacenaremos en un array de objetos del tipo DirectoryInfo los subdirectorios correspondientes a dicho directorio. Mediante un bucle recorreremos dicho array, y por cada elemento del mismo crearemos un objeto del tipo Tema que almacenará en su propiedad _Nombre el nombre del directorio en cuestión. De esta forma obtenemos un listado de objetos del tipo Tema, cada uno de los cuales estará relacionado con la correspondiente subcarpeta del directorio App_Themes.

Por lo tanto, definiremos una propiedad de perfil, a la que denominaremos Tema, y que resultará de la siguiente manera:

```
<profile>
        <properties>
                <add name="Tema" defaultValue="Azul" type="String"/>
        </properties>
</profile>
```

Como vemos, dicha propiedad tendrá un valor por defecto: el tema Azul. Seguidamente pasaremos a configurar la interfaz que permitirá al usuario elegir entre cada uno de los temas. Primeramente insertaremos un objeto ObjectDataSource (ODS) en la plantilla

LoggedInTemplate del control LoginView de la página maestra y elegiremos como método correspondiente a la operación SELECT el método GetTemas(), creado anteriormente en la clase BLLTema.

A continuación añada un control DropDownList cuyo origen de datos será dicho ObjectDataSource, así como un botón para confirmar nuestra elección.

```
<table>
        <tr>
          <td style="width: 35px" valign="middle">
            <asp:Label ID="Label2" runat="server" CssClass="textos" Text="Tema: "
SkinID="titulosmaster"></asp:Label></td>
          <td>
            <asp:DropDownList ID="Ddl_tema" runat="server" CssClass="textos"
DataSourceID="ODS"
              DataTextField="_Nombre" DataValueField="_Nombre">
            </asp:DropDownList></td>
        </tr>
      </table>
<asp:Button ID="Button1" runat="server" CssClass="textoscenter"
OnClick="Cambiar_Tema"
        Text="Aceptar" /><br />
      <br />
```

Por último, agregaremos en el manejador del evento load del control Login.ascx el siguiente código:

```
protected void Page_Load(object sender, EventArgs e)
  {

    if (!IsPostBack && HttpContext.Current.User.Identity.IsAuthenticated)
    {
      ((DropDownList)LoginView1.FindControl("Ddl_tema")).Text = Profile.Tema;
    }
  }
```

De esta manera cuando el usuario inicie sesión el valor del DropDownList será igual al valor que tenga la propiedad Tema del perfil del usuario. Por otro lado, definiremos un manejador para el evento Clic del botón aceptar, cuyo código es el siguiente:

```
protected void Cambiar_Tema(object sender, EventArgs e)
  {
    Profile.Tema = ((DropDownList)LoginView1.FindControl("Ddl_tema")).SelectedValue;
    //Guardar el tema escogido por el USR
    Profile.Save();
```

```
        //Aplicar el tema inmediatamente
        Server.Transfer(Request.FilePath);
    }
```

Como vemos, el manejador almacena en la propiedad Tema del perfil el valor seleccionado en el DropDownList. Y una vez actualizados los datos del perfil, haremos una recarga de la página actual para que los cambios se vean reflejados.

Para poder cambiar de tema dinámicamente y que éste se aplique a todas y cada una de las páginas del sitio web, crearemos una clase de la que heredarán todas ellas, denominada BasePage.cs. Esta clase heredará a su vez de la clase System.Web.UI.Page.

```
public class BasePage : System.Web.UI.Page
{
    protected override void OnPreInit(EventArgs e)
    {
        base.OnPreInit(e);
        System.Web.Profile.ProfileBase perfil = new System.Web.Profile.ProfileBase();
        perfil = HttpContext.Current.Profile;
        Page.Theme = perfil.GetPropertyValue("Tema").ToString();
    }
}
```

6.18.3. Propiedades de los temas

Como hemos mencionado anteriormente, para cada tema existirá una carpeta en cuyo interior insertaremos una hoja de estilos, Hoja_Estilos.css, un archivo de máscaras, SkinFile.skin, y una carpeta denominada Imágenes que contendrá las imágenes usadas en cada tema. Cada tema se diferencia del resto fundamentalmente, en la combinación de colores. Por ejemplo en el Tema Rojo, el color de fondo de todos los paneles incluidos en la página maestra será el LightCoral. Los títulos, o etiquetas por ejemplo tendrán en el caso del tema Rojo un Color de Fuente igual a Maroon. Además tanto el logo de la cabecera como el menú principal, los DataList y los GridView tendrán una apariencia en dichos tonos.

Para lograr este efecto, emplearemos por un lado la hoja de estilos y por otro el archivo de máscaras. La hoja de estilos definirá el estilo del menú principal y del menú admin, mientras que el archivo de máscaras contendrá diferentes propiedades para ciertos controles. Es el caso,

de por ejemplo, la imagen del logo, los paneles o los calendarios. Dichas propiedades variarán de un tema a otro.

```
<asp:Hyperlink runat="server" SkinId = "titulosmaster" ForeColor = "Maroon"/>
<asp:Linkbutton runat="server" SkinId = "titulosmaster" ForeColor = "Maroon"/>
<asp:Image runat="server" SkinId = "Imagenes" BorderColor = "LightCoral"/>
<asp:Image runat="server" SkinId = "Logo" ImageUrl="Imagenes/Logo.png"/>
<asp:Panel runat="server" SkinId = "Panel" BackColor = "LightCoral" BorderColor="Maroon"/>
<asp:Label runat="server" SkinId = "titulosmaster" ForeColor = "Maroon"/>
<asp:Gridview runat="server" SkinId="GridView">
<RowStyle BackColor="#FFF0F0" />
<EditRowStyle BackColor="#FFF8FF" />
<SelectedRowStyle BackColor="MistyRose" ForeColor="#333333" Font-Bold="True" />
<PagerStyle BackColor="RosyBrown" ForeColor="White" CssClass="textos"
HorizontalAlign="Right" />
<HeaderStyle BackColor="RosyBrown" ForeColor="White" CssClass="textosnegrita"
HorizontalAlign="Center" Font-Size="11px" Font-Bold="True"/>
<AlternatingRowStyle BackColor="#FFF8FF"/>
</asp:GridView>
<asp:DetailsView  runat="server" SkinID="Datalist" BorderColor="RosyBrown"
ForeColor="#333333" >
<FooterStyle BackColor="MistyRose" ForeColor="White" Font-Bold="True"></FooterStyle>
<CommandRowStyle BackColor="MistyRose" Font-Bold="True"></CommandRowStyle>
<EditRowStyle BackColor="#FFF8FF"></EditRowStyle>
<RowStyle BackColor="#FFF8FF"></RowStyle>
<PagerStyle BackColor="#FFF0F0" ForeColor="White" HorizontalAlign="Center"></PagerStyle>
<FieldHeaderStyle BackColor="#FFF0F0" Font-Bold="True"></FieldHeaderStyle>
<HeaderStyle BackColor="RosyBrown" ForeColor="White" Font-Bold="True"></HeaderStyle>
<AlternatingRowStyle BackColor="White"></AlternatingRowStyle>
</asp:DetailsView>
<asp:Calendar runat="server" SkinId = "CalenMaster" BackColor="White"
BorderColor="LightSalmon"  ForeColor="#400000">
<SelectedDayStyle BackColor="#009999" Font-Bold="True" ForeColor="Black" />
     <SelectedDayStyle BackColor="#009999" Font-Bold="True" ForeColor="Red" />
        <TodayDayStyle BackColor="#804040" ForeColor="White" />
        <SelectorStyle BackColor="#99CCCC" ForeColor="#336666" />
        <WeekendDayStyle BackColor="RosyBrown" />
        <OtherMonthDayStyle ForeColor="#999999" />
        <NextPrevStyle  ForeColor="Maroon" />
        <DayHeaderStyle BackColor="#804040" ForeColor="Bisque" Height="1px" />
        <TitleStyle BackColor="Bisque" BorderColor="LightSalmon" Font-Bold="True"
ForeColor="#400000"  />
</asp:Calendar>
```

Con respecto al resto de archivos de máscara, únicamente deberá variar los colores, para hacerlos amigables al usuario.

CONCLUSIONES

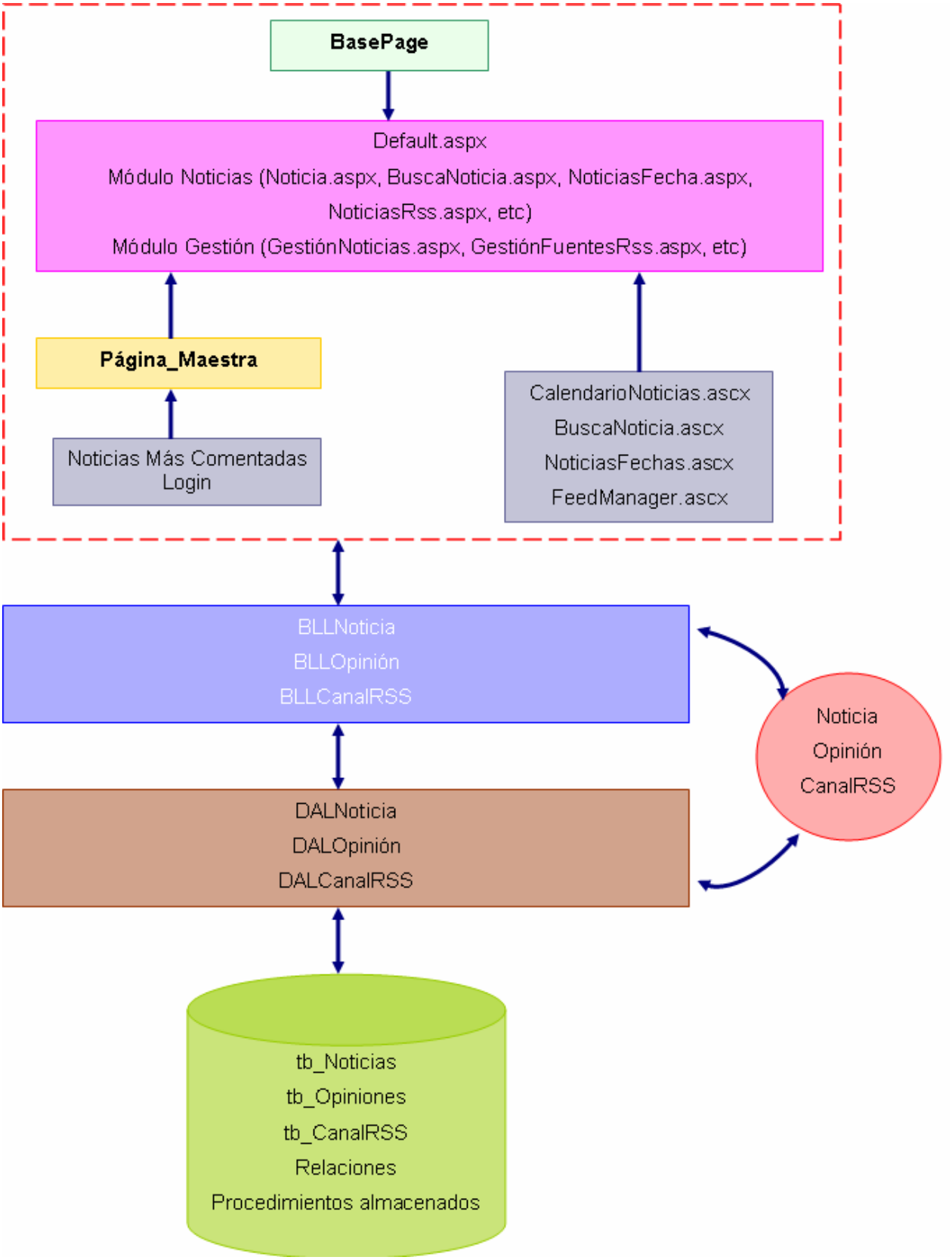
Con este proyecto se logró fortalecer el Pénsul académico de las carreras del Departamento de Computación, pues se detallaron los aspectos concernientes a la planificación del componente Programación Orientada a la Web II, con alto contenido práctico, pero sin dejar a un lado las bases teóricas fundamentales para el desarrollo del Portal.

En primer lugar se desarrolló la Aplicación Web para dar paso al diseño del componente Programación Orientada a la Web II.

Basado en la experiencia propia y en la cantidad de horas de las que dispone el componente curricular, el desarrollo completo de la aplicación Portal de Noticias servirá para dar al estudiante una visión amplia de lo que es, la programación orientada a la Web, con ASP.NET.

En este sentido, las prácticas de laboratorio estarán orientadas al desarrollo básico de la aplicación, sin dejar por ello un vacío en el estudiante, pues por la arquitectura utilizada, hace que el Portal de Noticias tenga módulos cuyo diseño e implementación sea repetitivo.

Por todos los factores antes expuesto y tratando de implementar los conocimientos básicos e imprescindibles en el estudiante, se dedujo como arquitectura básica del Portal de Noticias la siguiente:


BasePage
Default.aspx
Módulo Noticias (Noticia.aspx, BuscaNoticia.aspx, NoticiasFecha.aspx, NoticiasRss.aspx, etc)
Módulo Gestión (GestiónNoticias.aspx, GestiónFuentesRss.aspx, etc)
Página_Maestra
Noticias Más Comentadas
Login
CalendarioNoticias.ascx
BuscaNoticia.ascx
NoticiasFechas.ascx
FeedManager.ascx
BLLNoticia
BLLOpinión
BLLCanalRSS
Noticia
Opinión
CanalRSS
DALNoticia
DALOpinión
DALCanalRSS
tb_Noticias
tb_Opiniones
tb_CanalRSS
Relaciones
Procedimientos almacenados

BIBLIOGRAFÍA

Libros consultados:

- Ceballos Sierra Francisco Javier (2006), “Enciclopedia Microsoft Visual C#, España, Editorial RA-MA.
- McClure Wallace B., Scout Cate, Paul Glavich, Graig Shoemaker (2006), “Fundamentos Ajax con ASP .NET”, España, Editorial ANAYA Multimedia.

Planes docentes consultados:

- Espinoza Hernández Denis (2007), “Plan docente para la asignatura de Aplicaciones Telemáticas”, UAH – España.
- Medina Rodríguez Valeria (2007), “Plan docente para la asignatura de Seguridad en Redes”, UAH – España.

Referencias a Internet consultadas:

- http://msdn.microsoft.com/es-es/library/default.aspx
- http://www.scourdesign.com/
- http://peterkellner.net/
- http://www.subgurim.net/
- http://www.desarrolloweb.com/
- http://hanzcocchi.net/instalar-y-configurar-asp-net-ajax/
- http://fckeditor.uptodown.com/
- http://www.mentores.net/Default.aspx?tabid=104&type=art&site=71&parentid=34
- http://www.lajornadadeoriente.com.mx/rss/rss.php
- http://es.geocities.com/rss_guia_facil/como_crear_rss.html
- http://msdn.microsoft.com/es-es/library/wcyt4fxb(VS.80).aspx

ANEXOS

1. INSTALACIÓN Y CONFIGURACIÓN DE ASP .NET AJAX

Para empezar a usar la versión de Microsoft ASP.NET AJAX debemos de tener en cuenta que es un framework separado de la versión actual de Microsoft Visual Studio 2005. De modo que para trabajar con dicha plataforma debemos descargar dos componentes adicionales:

1. **Microsoft ASP .Net 2.0 AJAX Extensions 1.0**. Es el framework que permite utilizar la tecnología AJAX en nuestros proyectos de ASP .Net. Con esto sería suficiente para comenzar a trabajar con AJAX, sin embargo para añadir mayor funcionalidad también conviene instalar el siguiente.

2. **AJAX Control Toolkit**. Es un conjunto de controles ya desarrollados y listos para ser usados que permiten utilizar la tecnología de ASP.NET AJAX. Estos controles estarán disponibles en el cuadro de controles del usuario listos para arrastrar y usar. Podemos ver una demostración de los mismos en la siguiente url: http://ajax.asp.net/ajaxtoolkit/

Ambos instalables los podemos encontrar en la url oficial de ASP.NET AJAX: http://www.asp.net/ajax/downloads/

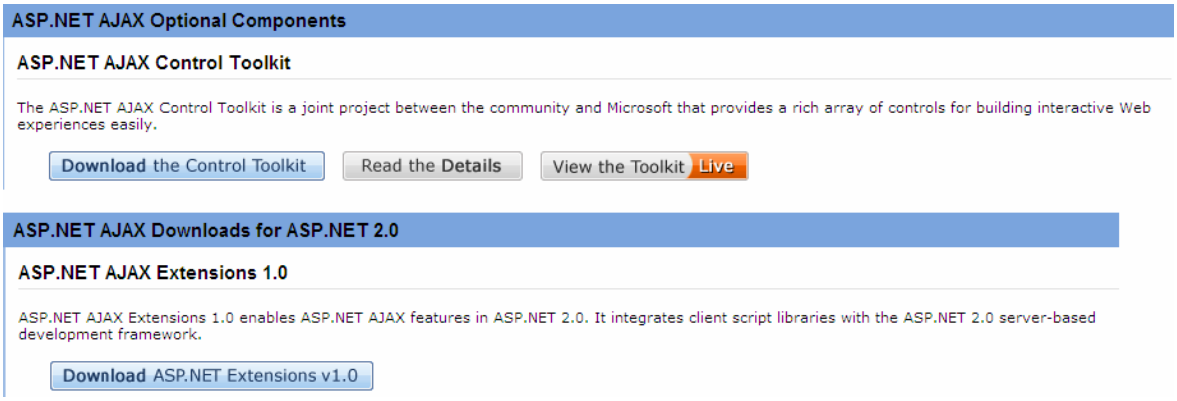


Fig. 110 Instalación de ASP .NET AJAX.

Una vez que hayamos descargado ambos componentes debemos de seguir los siguientes pasos:

1. Instalamos el archivo ASPAJAXExtSetup.msi correspondiente al primer componente descargado. Este contiene el framework de AJAX. Se trata de una instalación típica en la que bastará con ir pulsando siguiente. Por defecto se instalará en la carpeta C:\Archivos de programa\Microsoft ASP.NET\ASP.NET 2.0 AJAX Extensions de nuestro PC. Dentro de esta carpeta se creará la subcarpeta v1.061025 que corresponde a la versión actual.

   Como se ha mencionado anteriormente, ya podríamos empezar a trabajar con ASP.NET AJAX. Sin embargo, no dispondríamos de los controles del propio framework de AJAX, el componente conocido como AJAX Control Toolkit.

2. A continuación descomprimimos el archivo AjaxControlToolkit.zip dentro de la misma carpeta donde hemos instalado AJAX, en este caso en C:\Archivos de programa\Microsoft ASP.NET\ASP.NET 2.0 AJAX Extensions, y dentro de una carpeta de nombre AjaxControlToolkit. Una vez descomprimido veremos una solución con el nombre AjaxControlToolKit.sln

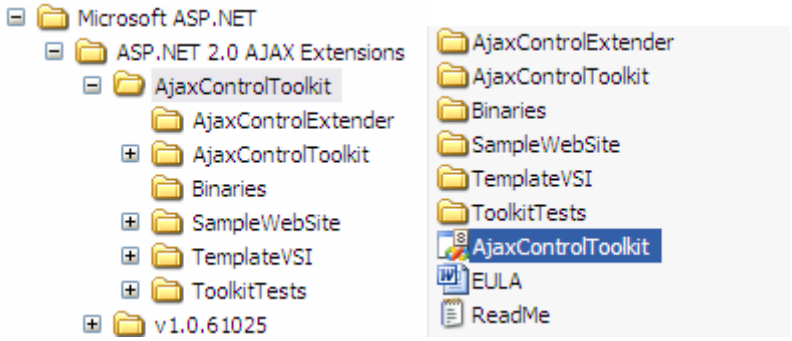


Fig. 111 AjaxControlToolkit.sln

3. Abrimos la solución, la cual contiene varios proyectos y generamos el denominado TemplateVSI, para ello hacemos clic con el botón derecho del ratón sobre el proyecto (no sobre la solución) y seleccionamos Generar. Una vez hecho esto, procedemos a ejecutar la solución, lo que cargará la página de ejemplo de los controles y hará que se compile el proyecto generándose las librerías que necesitamos.

4. Seguidamente, abrimos la carpeta C:\Archivos de programa\Microsoft ASP.NET\ASP.NET 2.0 AJAX Extensions\AjaxControlToolkit\SampleWebSite\Bin donde encontraremos dos archivos con el mismo nombre (AjaxControlToolKit) uno con extensión DLL y el otro con extensión PDB. Copiamos estos archivos y los pegamos en la carpeta C:\Archivos de programa\Microsoft ASP.NET\ASP.NET 2.0 AJAX Extensions\AjaxControlToolkit\Binaries.

5. A continuación ingresamos en la carpeta C:\Archivos de programa\Microsoft ASP.NET\ASP.NET 2.0 AJAX Extensions\AjaxControlToolkit\TemplateVSI\bin y hacemos doble clic en el archivo AjaxControlExtender.vsi. Este es un archivo de instalación, y en caso de que apareciera algún mensaje que nos pregunte si deseamos sobrescribir archivos, seleccionaremos Aceptar.

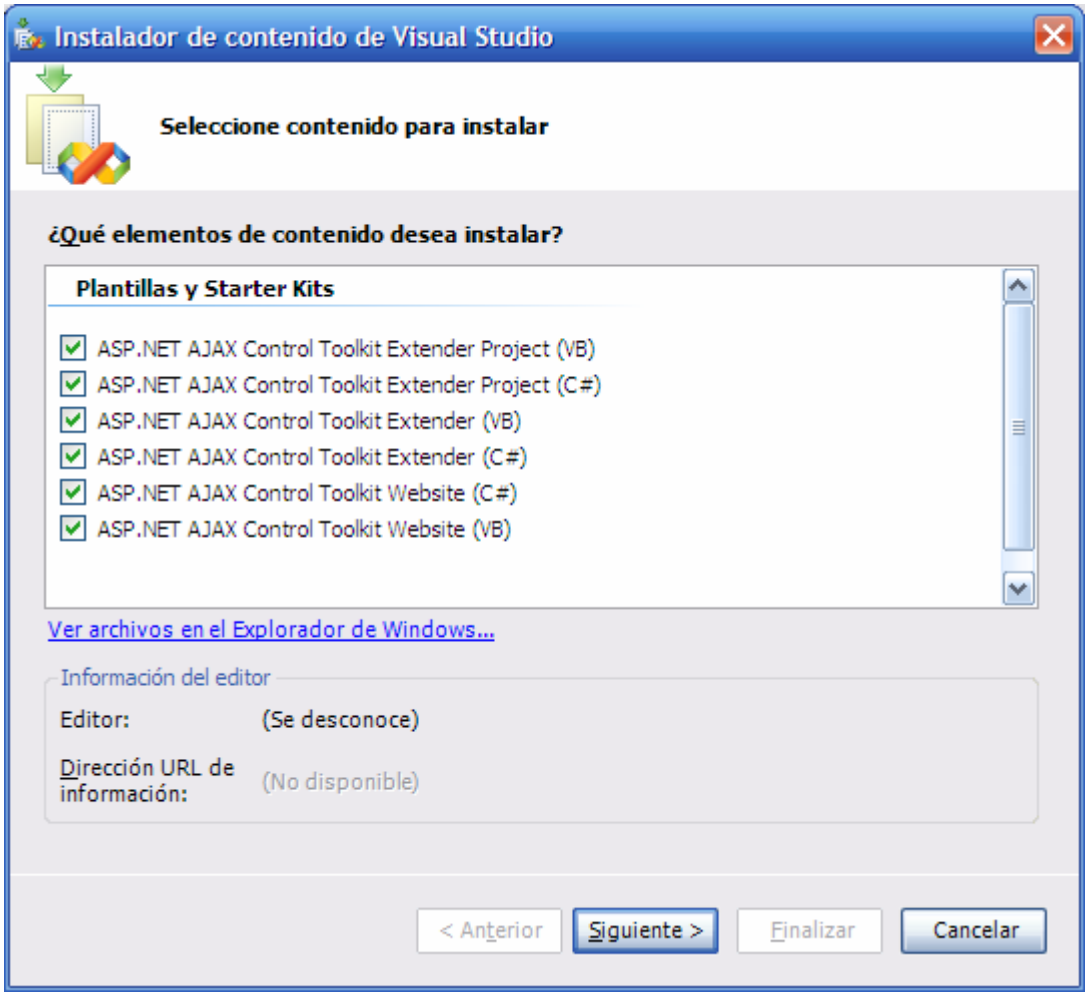


Fig. 112 Instalando Ajax Control Toolkit.

6. Cerramos todas las ventanas de Windows, abrimos el Visual Studio 2005 y creamos un nuevo sitio Web seleccionando la plantilla Ajax Control Toolkit Web Site. Como vemos, también tenemos disponible la plantilla ASP.NET AJAX- Enabled Web Site.

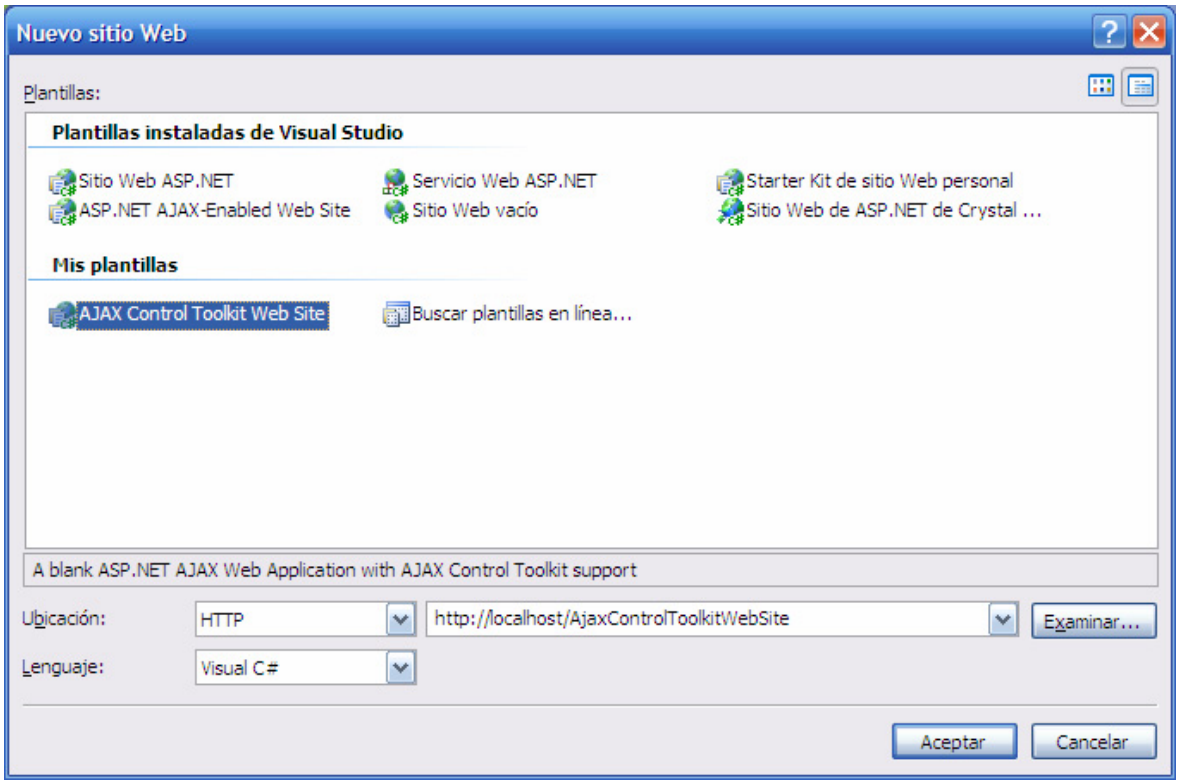


Fig. 113 Creando el sitio Web.

7. La principal diferencia entre uno y otro, es que el primero contiene la librería AjaxControlToolkit en su carpeta bin por defecto. Si seleccionamos la plantilla ASP.NET AJAX- Enabled Web Site para nuestro proyecto o queremos añadir el AjaxControlToolkit a un proyecto existente de este tipo, deberíamos hacer lo siguiente: agregaríamos una referencia al proyecto haciendo clic con el botón derecho sobre el mismo y seleccionando agregar referencia.

   Una vez que se nos abriese la ventana Agregar referencia, seleccionaríamos la pestaña Examinar, ubicaríamos el archivo AjaxControlToolkit.dll que se encuentra en la carpeta C:\Archivos de programa\Microsoft ASP.NET\ASP.NET 2.0 AJAX Extensions\AjaxControlToolkit\Binaries y haríamos clic en aceptar.

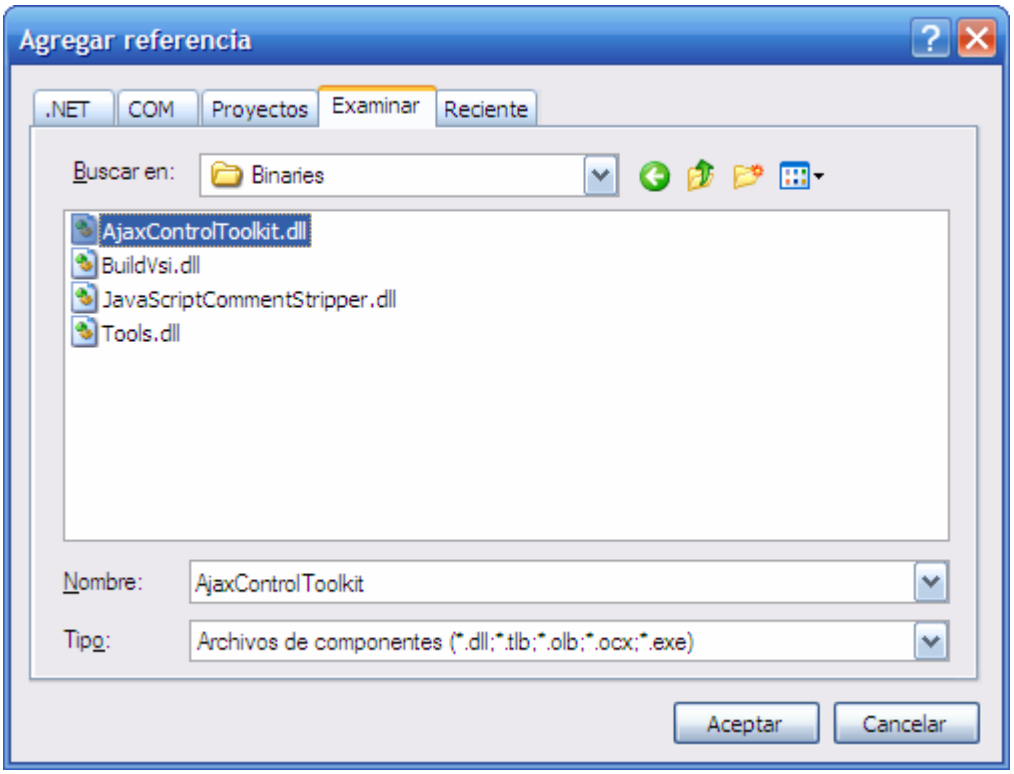


Fig. 114 Agregar la referencia AjaxControlToolkit.dll

Otra manera de proceder sería copiar directamente el archivo .dll a la carpeta bin de nuestro proyecto.

8. Una vez creado el sitio Web, nos daremos cuenta que en la parte inferior del cuadro de herramientas hay una nueva ficha llamada AJAX Extensions. Pero todavía no aparecen los controles del Ajax Control Toolkit.

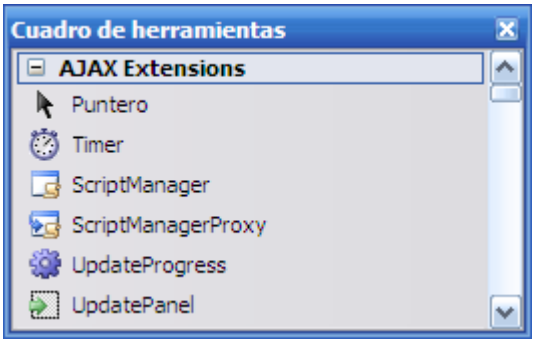


Fig. 115 Controles AJAX.

Para agregarlos, hacemos clic derecho sobre el cuadro de herramientas, seleccionamos la opción Agregar Ficha y le damos el nombre de Ajax Control Toolkit. Sobre la ficha que acabamos de crear, hacemos clic derecho y seleccionamos la opción Elegir Elementos.

9. En la ventana de Elegir Elementos hacemos clic en examinar y ubicamos el archivo AjaxControlToolkit.dll que se encuentra en el directorio C:\Archivos de programa\Microsoft ASP.NET\ASP.NET 2.0 AJAX Extensions\AjaxControlToolkit\Binaries.

10. Hacemos clic en Aceptar y ya podemos ver nuestros controles del Ajax Control Toolkit en el cuadro de Herramientas. La mayor ventaja es que a partir de ese momento, no necesitamos repetir estos pasos otra vez cada vez que creemos un nuevo site del tipo Ajax Control Toolkit Web Site.

2. EDITOR DE TEXTO: FCKEDITOR

2.1. Introducción

FCKeditor es un editor de texto HTML de código abierto que proporciona muchas de las poderosas funcionalidades de conocidos editores, como MS Word, a la web. Es realmente ligero y no requiere ningún tipo de instalación en el ordenador del cliente.

Es una aplicación JavaScript que se ejecuta en el navegador. Se puede utilizar sin ningún tipo de integración en el lado del servidor, pero la manera más clara y eficiente de utilizarlo es eligiendo un paquete de integración en el lado del servidor que se adapte a nuestras necesidades.

En este caso para ASP.NET está disponible un paquete que posibilita el uso de FCKEditor como otro control en nuestros formularios Web. Dicho paquete se llama FCKeditor.Net

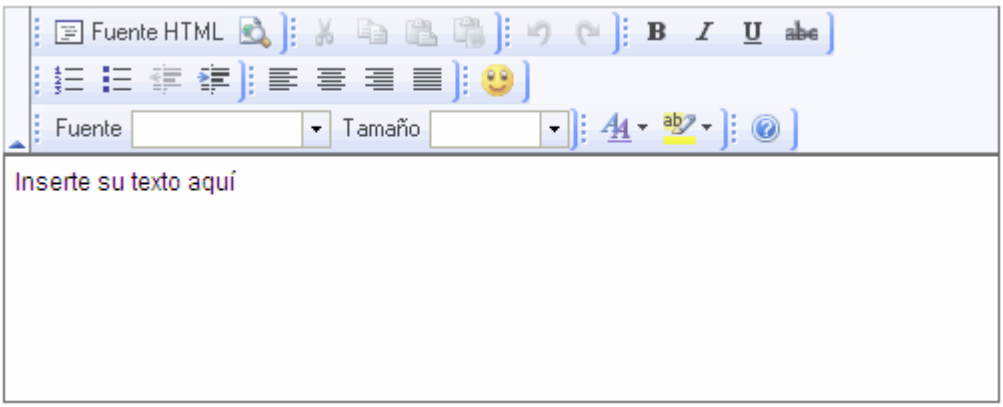


Fig. 116 Editor de texto: FCKeditor.Net

2.2. Instalación

La instalación del editor en el portal Web es muy sencilla. Primero instalaremos el paquete genérico y después el correspondiente a ASP.NET.
Tenemos que seguir los siguientes pasos:

- Descargar la última versión del editor en la siguiente url: http://sourceforge.net/project/showfiles.php?group_id=75348

- Descomprimir el fichero en un directorio llamado “FCKeditor” en la raíz de nuestro sitio web.

Este paquete contiene una página de ejemplos disponible para cada lenguaje de programación soportado por el editor. Se encuentra en el directorio “_samples”. Para probar la instalación podemos escribir la siguiente url en nuestro navegador:
http:///(nuestro sitio web)/(path de instalación FCKeditor)/samples/default.html

Como hemos mencionado una vez instalado este paquete hay que proceder a instalar el correspondiente a ASP.NET. Para ello necesitamos vamos a realizar los siguientes pasos:

- Descargar la ultima versión del control ASP.Net en la siguiente url: http://sourceforge.net/project/showfiles.php?group_id=75348&package_id= 137125

- El archivo ZIP contiene el código fuente del control y una versión compilada del mismo, el archivo “bin/Release/ FredCK.FCKeditorV2.dll”. Como no vamos a hacer ningún cambio en el código fuente solamente tendremos en cuenta el fichero .dll, creando una referencia del mismo en el proyecto. Para esto tenemos dos opciones:
    - Copiar de forma natural el archivo a nuestro directorio bin del portal web.

    - Pulsando el botón derecho encima del directorio raíz del proyecto y en el menú contextual pulsar “Añadir referencia”. En la nueva pantalla seleccionaríamos el archivo “FredCK.FCKeditorV2.dll” del directorio donde lo tengamos guardado.

- Incluimos el control en la barra de controles del Visual Studio. Para ello con el botón derecho sobre la barra seleccionamos “Elegir Elementos”. Seguidamente pulsaremos el botón examinar y seleccionaremos el archivo “FredCK.FCKeditorV2.dll” del directorio bin del proyecto.

2.3. Configuración

El editor ya está listo para ser usado en nuestro portal. Para añadirlo a una página ASP.NET tenemos dos opciones:

- Arrastrar y soltar el control en la página deseada desde la barra de herramientas.
- Incluir la siguiente línea en la parte de arriba de la página fuente:

```
<%@ Register Assembly="FredCK.FCKeditorV2" Namespace="FredCK.FCKeditorV2"
TagPrefix="FCKeditorV2" %>
```

Seguidamente añadiremos el tag del editor dentro de un formulario.

```
<FCKeditorV2:FCKeditor id="FCKeditor1" runat="server" Basepage =
"/fckeditor"></FCKeditorV2:FCKeditor>
```

Debemos tener en cuenta que si estamos en una página derivada de la página maestra no tendremos que volver a añadir el formulario, esto daría lugar a un error de compilación. La propiedad BasePath del control contiene el path del directorio donde los scripts del editor han sido copiados. Y estableceremos la propiedad ValidateRequest de dicha página a "False"

Apariencia

El control FCKEditor nos ofrece la posibilidad de variar su aspecto de diferentes modos. Por un lado disponemos de tres skins distintos (default, office2003 y silver) que podemos elegir con la propiedad FCKConfig.SkinPath, para lo cual accederemos al archivo “fckconfig.js” del directorio “FCKEditor” de nuestro proyecto.

```
FCKConfig.SkinPath = FCKConfig.BasePath + 'skins/office2003/' ;
```

Por otro lado, existen dos tipos predefinidos de barras de herramientas para el editor, “Default” y “Basic”. Dependiendo del que queramos utilizar, haremos que el valor de la propiedad ToolBarSet del control FCKEditor sea uno u otro.

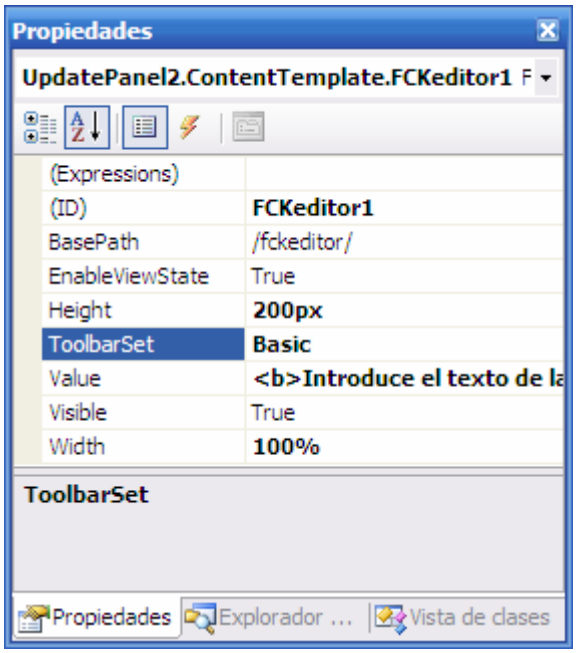


Fig. 117 ToolbarSet de FCKeditor.

La siguiente línea correspondería con el tipo “Basic”:

```
FCKConfig.ToolbarSets["Basic"] = [
['Bold','Italic','-','OrderedList','UnorderedList','-',
'Link','Unlink','-','About']
] ;
```

Mientras que el código siguiente pertenecería al tipo “Default”:

```
FCKConfig.ToolbarSets["Default"] = [
        ['Source','DocProps','-','Save','NewPage','Preview','-','Templates'],
        ['Cut','Copy','Paste','PasteText','PasteWord','-','Print','SpellCheck'],
        ['Undo','Redo','-','Find','Replace','-','SelectAll','RemoveFormat'],
        ['Form','Checkbox','Radio','TextField','Textarea','Select','Button','ImageButton','HiddenField
        '],
        '/',
        ['Bold','Italic','Underline','StrikeThrough','-','Subscript','Superscript'],
        ['OrderedList','UnorderedList','-','Outdent','Indent','Blockquote'],
        ['JustifyLeft','JustifyCenter','JustifyRight','JustifyFull'],
        ['Link','Unlink','Anchor'],
        ['Image','Flash','Table','Rule','Smiley','SpecialChar','PageBreak'],
        '/',
        ['Style','FontFormat','FontName','FontSize'],
        ['TextColor','BGColor'],
        ['FitWindow','ShowBlocks','-','About']
] ;
```

En nuestra aplicación personalizaremos el tipo “Basic” de la siguiente manera:

```
FCKConfig.ToolbarSets["Basic"] = [
        ['Source', 'Bold','Italic','StrikeThrough', 'TextColor','BGColor', '-
','OrderedList','UnorderedList','-','Link','Unlink','-','About']
] ;
```

Además añadiremos un nuevo tipo, denominado “Foro” que será de la siguiente forma: (Ver **Fig. 116).**

```
FCKConfig.ToolbarSets["Foro"] = [
        ['Source','Preview'],
        ['Cut','Copy','Paste','PasteText'],
        ['Undo','Redo'],
        ['Bold','Italic','Underline','StrikeThrough'],
        '/',
        ['OrderedList','UnorderedList','Outdent','Indent'],
        ['JustifyLeft','JustifyCenter','JustifyRight','JustifyFull'],
        ['Smiley'],
        '/',
        ['FontName','FontSize'],
        ['TextColor','BGColor'],
        ['About']
] ;
```

Por otro lado, personalizaremos los tipos de fuentes disponibles mediante la opción FCKConfig.FontNames y restringiremos los tamaños de las mismas mediante la opción FCKConfig.FontSizes.

```
FCKConfig.FontNames = 'Arial;Comic Sans MS;Courier New;Tahoma;Times New
Roman;Verdana' ;
FCKConfig.FontSizes = 'smaller;larger;xx-small;x-small;small;medium;large;x-large;xx-large' ;
```

Además cambiaremos las imágenes correspondientes a los iconos, así como el aspecto de la ventana que los muestra con las siguientes propiedades:

```
FCKConfig.SmileyPath = FCKConfig.BasePath + 'images/smiley/msn/' ;
FCKConfig.SmileyImages =
['regular_smile.gif','sad_smile.gif','wink_smile.gif','teeth_smile.gif','confused_smile.gif','tounge_smil
e.gif','embaressed_smile.gif','omg_smile.gif','whatchutalkingabout_smile.gif','angry_smile.gif','ange
l_smile.gif','shades_smile.gif','devil_smile.gif','cry_smile.gif','lightbulb.gif','thumbs_down.gif','thumb
s_up.gif','heart.gif','broken_heart.gif','kiss.gif','envelope.gif'] ;
FCKConfig.SmileyColumns = 8 ;
FCKConfig.SmileyWindowWidth = 320 ;
FCKConfig.SmileyWindowHeight = 210 ;
```

3. CONSTRUYENDO UN MOTOR DE BÚSQUEDA PARA EL SITIO WEB UTILIZANDO EL WEB SERVICE DE GOOGLE

Google provee un Web Service para realizar búsqueda a través de la base de datos de Google, con este servicio se puede brindar a un sitio la funcionalidad de un motor de búsqueda al estilo "Google …" , lo que representa una base de datos, disponible y fácil de manejar

3.1. Al API de Google

La información sobre el Web Service de Google la puedes encontrar en http://google.com/apis/. En esta dirección tendrás que bajar el Google Web API Developer's kit(666K) este kit incluye el archivo WSDL (Web Service Description Laguage) que describe el Web Service, y además brinda ejemplos de cómo acceder a este servicio a través de VB.NET o C# o Java.

Luego de que bajes el Google Web API Developer's Kit, necesitarás crear una cuenta con Google. Una vez creada la cuenta, se te asignará un número de licencia único. Esta licencia la podrás utilizar cada vez que el método de búsqueda sea invocado, con la única limitación de que solamente puedes realizar 1.000 llamadas al método de búsqueda por día.

3.2. Creando la Clase Proxy

Ahora el siguiente paso es crear una clase tipo "proxy" que se usará para llamar al Web service. Para realizar esto, primero necesitamos tener al WSDL file, este es un archivo XML que describe los servicios que el Web Service de Google tiene. Este archivo, *GoogleSearch.wsdl* está dentro del Google Web API Developer's Kit.

Si se utiliza Visual Studio .NET, copiamos el archivo al directorio de la aplicación ASP.NET (como: C:\Inetpub\wwwrooot\PortalWeb\). Entonces, en el Visual Studio .NET, Agregamos una Referencia Web (Web Reference) al proyecto. En el cuadro de diálogo se ingresa el URL del archivo WSDL, el cual debería ser algo así: *http://localhost/PortalWeb/GoogleSearch.wsdl.* Para completar el proceso, hacemos clic en Add Reference. Esto creará la clase proxy usando el namespace *localhost* (luego puedes cambiarle el nombre si quieres).

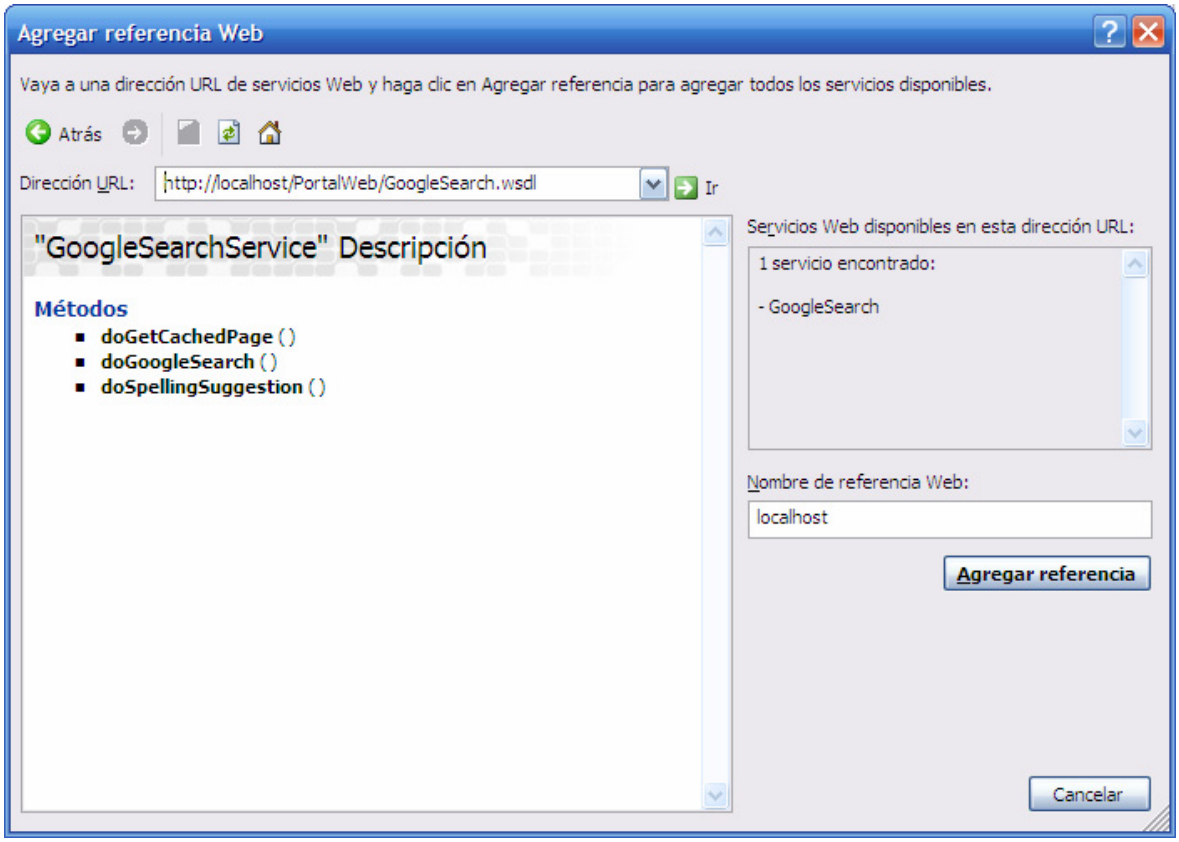


Fig. 118 Agregar una referencia Web.

3.3. Método doGoogleSearch()

Ahora que ya está creada la clase proxy, llamar al Web Service a través de una página SP.NET es súper sencillo. Pero antes de hacerlo examinemos qué parámetros espera el Service que le enviemos. Afortunadamente estos parámetros de entrada están detallados en la sección de referencia en el sitio de Google. Como en este artículo nos referiremos a realizar una búsqueda vía este servicio, solamente examinemos los parámetros para el método doGoogleSearch(). Este método recibe 10 parámetros:

- **key**: Provista por Google, es requerida para acceder al servicio de Google. Se utiliza esta credencial con fines de autenticación y autorización.
- **q:** Lo que se va a buscar en sí, más información en la referencia de google.
- **start:** El índice “Zero-based“ del primer resultado deseado.

- **maxResults:** El número de resultados deseados por consulta. El valor máximo por consulta es 10. N**ota:** si existe una consulta que no tiene tantas concordancias, el número actual de resultados que tendrás podría ser menor que lo que requieras.
- **filtro:** Activa o desactiva el filtro automático de resultados, el cual excluye resultados muy similares o resultados que vengan del mismo Web host.
- **restricts:** Restringe la búsqueda a una parte de los sitios indexados por Google, como un país como “Ecuador“ o un tópico como “ASP.NET“ o inclusive un sitio como “javier-romero.com“. (Más detalles Restricts).
- **safeSearch:** un valor de tipo booleano que activa el filtro de resultados con contenido para adultos.
- **lr:** *Language Restrict* - Restringe la búsqueda a documentos de uno o más lenguajes.
- **ie:** *Input Encoding* - Este parámetro ha sido descontinuado y es ignorado. Todos los requerimientos a los APIs deben ser hechos con codificación UTF-8.
- **oe:** *Output Encoding* - Este parámetro ha sido descontinuado y es ignorado. Todos los requerimientos a los APIs deben ser hechos con codificación UTF-8.

El método *doGoogleSearch()* retorna una instancia del objeto *GoogleSearchResult*. Este objeto tiene una propiedad *resultElements*, el cual es un arreglo de objetos *ResultElement.* Cada objeto de este tipo tiene un número de propiedades como *title, snippet, URL, summary* y otros más. Ahora sólo lo debe utilizar, desde su página ASP .NET.

4. MANUAL DE INSTALACIÓN

A la hora de poner en marcha las funcionalidades de nuestro portal, debemos realizar un análisis desde dos perspectivas: por un lado, desde el lado del servidor, y por otro desde la perspectiva del usuario.

Para poder implantar nuestra aplicación en la máquina donde deseemos, necesitaremos de la instalación previa de, al menos, dos importantes herramientas: por un lado, el gestor de bases de datos Microsoft Sql Server, destinado a gestionar y almacenar toda la información de nuestro sitio; y por otro el servidor IIS de Microsoft, para albergar todas las páginas que conforman nuestro portal. Por otra parte, y desde el punto de vista de los usuarios que se dirijan a nuestro sitio Web, tan sólo será necesario que dispongan de cualquier navegador Web.

4.1. IIS (Internet Information Server)

Internet Information Server es el servidor de páginas Web de Windows. Se incluye en la versión Profesional de Windows XP. Es capaz de convertir a un ordenador en un servidor de Internet, por lo que podremos publicar páginas Web tanto local como remotamente (servidor Web) en aquellas máquinas dónde lo instalemos.

El servidor Web se basa en varios módulos que le dan capacidad para procesar distintos tipos de páginas, por ejemplo Microsoft incluye los de Active Server Pages (ASP) y ASP.NET.

Una de los aspectos más reseñables de IIS, es el ámbito de la seguridad. Conceptos como autenticación o autorización cobrarán un especial protagonismo a la hora de configurar el acceso a nuestra aplicación.

Existen otros servidores Web que pueden usarse como alternativa a éste, como por ejemplo Apache, Cherokee que son desarrollados mediante software libre.
Para su instalación hemos de agregar la aplicación ya que por efecto no suele venir incluida. Acudimos al Panel de control, y en su margen izquierdo pinchamos sobre Agregar o quitar componentes de Windows.

A continuación se nos mostrará una ventana en la que podremos seleccionar los elementos que queramos agregar. En la lista, marcamos la casilla Servicios de Internet Information Server. Por defecto aparecerán seleccionados varios componentes de los que ofrece IIS. Si clickeamos sobre detalles podremos elegir qué componentes deseamos instalar. Entre los que figuran se encuentran, extensiones de Frontpage, un servidor de FTP, servicios adicionales de IIS y hasta un servidor de SMTP para el envío de correos electrónicos.

Si desconocemos qué componentes debemos instalar, podemos dejar las opciones tal y como vienen marcadas en un principio, y agregar en un futuro aquello que necesitemos. Haciendo clic en siguiente, daremos fin a la instalación.

Para comprobar que hemos instalado con éxito nuestro servidor Web, podemos hacerlo introduciendo en el explorador de Internet http://localhost. De ser así, nos aparecerá una página confirmándolo y a su vez una ventana emergente en un lateral con toda la documentación relacionada, en caso de que la hubiéramos instalado.

El objetivo tras finalizar el proceso de instalación, es ubicar nuestro sitio Web en el servidor. En un primer momento, cuando accedemos a http://localhost, observamos que nos aparece un sitio Web predeterminado. Éste encuentra en el disco duro, concretamente en la carpeta C:/inetpub/wwwroot. Si accedemos a ella a través del explorador de Windows, nos encontraremos con los archivos que conforman el sitio, de entre los cuales cabe destacar el denominado iisstart.asp, que se pone en marcha cada vez que accedemos al sitio.

La manera de publicar nuestro sitio Web resulta ahora sencilla. Tan sólo tendremos que incluir todos nuestros archivos en la carpeta del sitio Web predeterminada. Si por ejemplo, agregamos un archivo llamado portada.aspx, bastará con teclear en nuestro explorador de Internet http://localhost/portada.aspx para poder visualizarlo.

Hemos de mencionar que estaríamos cometiendo un error si intentáramos acceder a nuestra página Web tecleando una ruta como la siguiente: C:/Inetpub/wwwroot/portada.aspx, pues de esta forma no se estaría pasando a través del servidor Web y la página no se ejecutaría.

En nuestro caso, y tratándose de un sitio más o menos complejo, sería conveniente crear una carpeta dentro del directorio wwwroot con el nombre de nuestro sitio e incluir dentro todos los archivos que lo compongan. En el explorador habríamos de teclear http://localhost/PortalWeb.

.

Si accedemos al sitio Web sin indicar el archivo que deseamos ejecutar, (http://localhost/PortalWeb) se mostrará la página por defecto. En IIS este archivo suele ser default.aspx o default.htm. Esto quiere decir que deberíamos incluir un archivo con alguno de estos dos nombres para que se muestre en caso de que el usuario no indique una página en particular.

Igualmente podríamos acceder a él tecleando la ruta completa http://localhost/PortalWeb/default.aspx

Administración de IIS

Para acceder a las propiedades de administración de IIS debemos seguir los siguientes pasos:

Pulsando con el botón derecho del ratón en MiPC, seleccionamos la opción *Administrar*. A continuación se nos abrirá una ventana denominada “Administración de equipos”. En la parte inferior de la lista de la izquierda aparece “Servicios y aplicaciones”, y dentro de los elementos que de despliegan, “Servicios de Internet Information Server”. Existen otros caminos para acceder a las opciones de administración, aunque ésta es de las más sencillas y usuales.

Una vez aquí, podremos configurar numerosos aspectos de nuestro servidor Web, desde la posibilidad de crear directorios virtuales a la opción de modificar las propiedades de configuración.

Cómo crear un directorio virtual

Cuando hablamos de directorio virtual no estamos refiriendo a un directorio que no está alojado en la carpeta de publicación por defecto de IIS, es decir un directorio fuera de wwwroot, pero que sin embargo si va a poder ser accedido mediante el servidor Web como si estuviera dentro de dicho directorio.

Para acceder a nuestro directorio virtual no tendremos más que concatenar su nombre al de la dirección de localhost, es decir http://localhost/directoriovirtual.

El directorio requerido puede hallarse en cualquier lugar de nuestro disco duro, e incluso en otro equipo perteneciente a la red. Para definir un directorio virtual, debemos pulsar con el botón derecho sobre el sitio Web donde queramos crearlo y seleccionar la opción “Nuevo>Directorio Virtual”. Será en este punto cuando se ejecute un asistente que nos irá guiando paso a paso en el proceso de configuración.

En primer lugar el asistente nos preguntará por el “alias” o nombre lógico que deseamos darle al directorio. A continuación nos solicitará la localización física de la carpeta en nuestro disco duro o red, y finalmente nos preguntará sobre los permisos que deseamos dar a ese directorio.

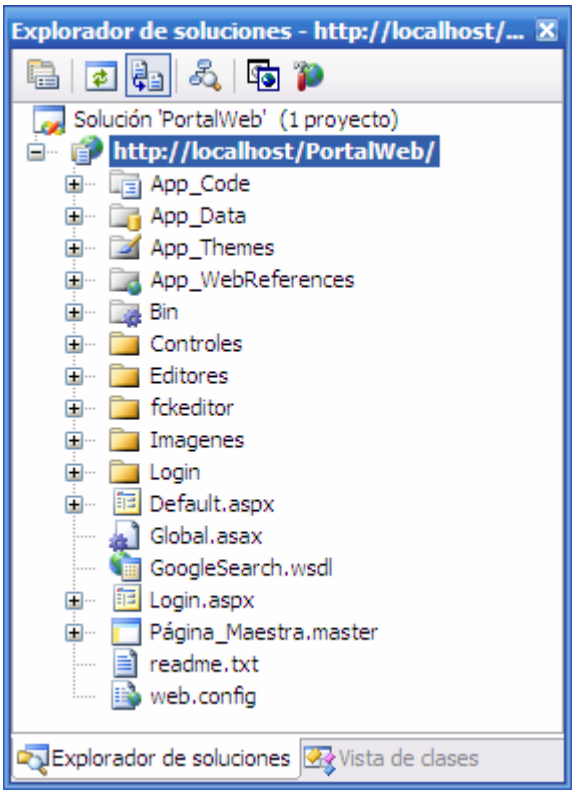
Será suficiente con habilitar los permisos de lectura y ejecución de sentencias de comandos (por ejemplo ASP) para una correcta configuración. Una vez finalizado el asistente, ya podemos hacer uso de nuestro directorio virtual con todas sus prestaciones.

Permisos de carpetas

Para el correcto funcionamiento de la aplicación deberemos hacer que el usuario ASPNET de nuestro equipo tengo permisos sobre la misma. Para ello, en la pestaña Seguridad de las propiedades de la carpeta en cuestión, agregaremos dicho usuario y le daremos permisos para modificar, leer y ejecutar, mostrar el contenido de la carpeta, leer y escribir.

5. ORGANIZACIÓN DEL SITIO WB

Un aspecto muy importante en el desarrollo de nuestro sitio Web es la organización de los elementos que lo componen.



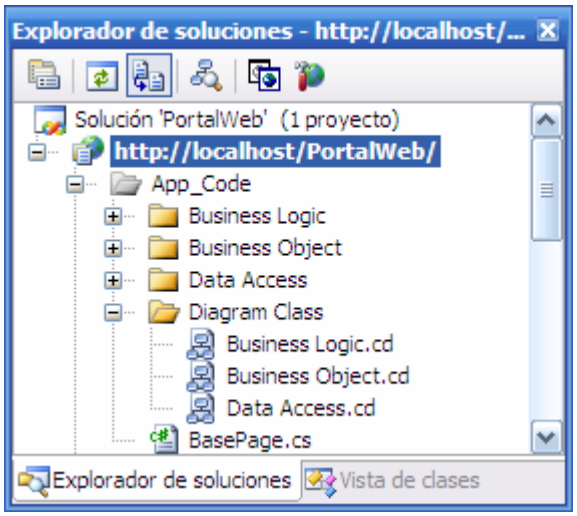
Para su estructuración nos apoyaremos en las carpetas predefinidas que nos ofrece Visual Studio para el desarrollo de cualquier aplicación Web. Asimismo, crearemos también nuestras propias carpetas para dar cabida a los distintos elementos, según se ha venido explicando a lo largo del desarrollo del sito.

La imagen del margen muestra con claridad la organización fundamental de nuestro sitio Web.

De la carpeta raíz subyacerán una serie de carpetas principales, que procedemos a analizar a continuación:

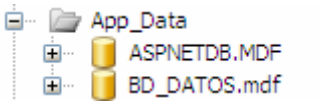
Carpeta App_Code

Será aquí donde se ubique el código referente a las diversas clases existentes. Tendremos una carpeta dedicada para los objetos de negocio en la que se ubicarán tanto las clases referentes a objetos individuales como las colecciones de objetos. En la carpeta Data Acces se encontrarán todas las clases de la capa DAL, y en la de Business Logic las de la BLL.



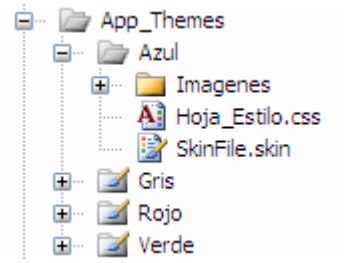
En la carpeta Diagrams Class se almacenarán los correspondientes diagramas de cada una de las carpetas anteriormente mencionadas. Por último, incluiremos también el código de clase BasePage.cs de la que se derivarán el resto de páginas de la web.

Carpeta App_Data



En esta carpeta se encontrarán las dos bases de datos que emplearemos en la aplicación: ASPNETDB.mdb para la gestión y administración del sitio Web; y BD_DATOS.mdf, para almacenar toda la información disponible en nuestro portal.

Carpeta App_Themes



Esta será la carpeta destinada a albergar todo lo referente a la funcionalidad de temas y skins.

Cada tema dispondrá de una carpeta, y en ella se incluirá su hoja de estilos css correspondiente, un archivo .skin y el conjunto de imágenes de las que se hará uso para ese tema.

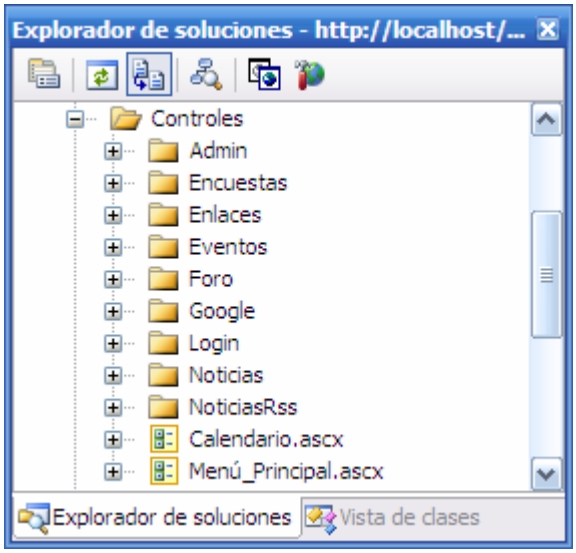
Carpeta Web_References

Esta será la carpeta que albergará la referencia Web que proporciona Google para implementar el motor de búsqueda del Portal.

Carpeta Bin

En la carpeta Bin se almacenarán algunas bibliotecas de los controles ajenos a ASP que instalaremos y emplearemos en la aplicación.

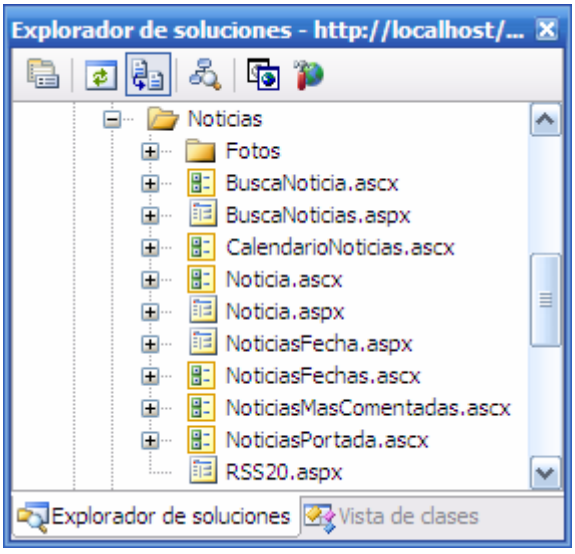
Carpeta Controles



Esta carpeta, creada por nosotros, albergará la mayoría de los controles y páginas que desarrollemos. Las clasificaremos por categorías en diferentes subcarpetas.

Algunos de los controles que constituyen la masterpage irán, sin embargo, fuera de estas subcarpetas. Como muestra, analizaremos el contenido de una de ellas.

Carpeta Noticias



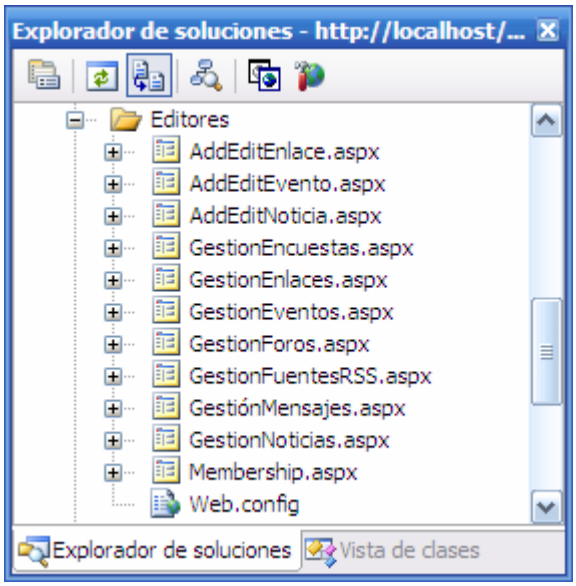
En la carpeta noticias incluiremos todos los archivos vinculados con la sección noticias. Como podemos apreciar, nos encontraremos tanto con páginas específicas como con los controles incluidos en ellas.

También crearemos aquí una carpeta destinada a albergar todas las imágenes que se vayan almacenando en la base de datos, y cuya ruta de acceso se guardará en el campo correspondiente de la propia base. Este mecanismo se repetirá también, por ejemplo, en la carpeta de Eventos.

Carpeta Editores

En la carpeta editores colocaremos todos archivos referentes a cada uno de los modos del modo admin. Asimismo, y debido a que su acceso sólo estará permitido para aquellos usuarios cuyo rol corresponda con el de “administradores”, se incluirá un pequeño fichero webconfig en el que se detallará dicha propiedad.

Carpeta fckeditor

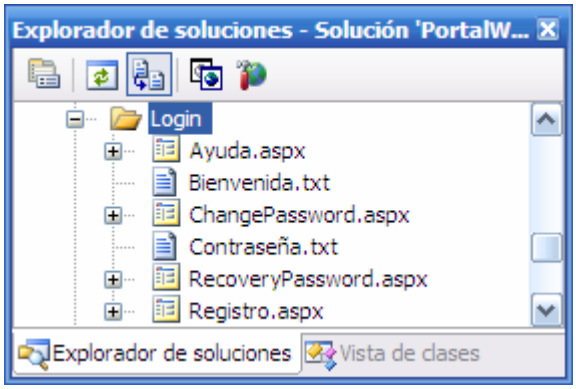


La siguiente carpeta es la del editor de textos fckeditor. En ella se encontrarán todos los archivos que lo componen, así como los correspondientes a las modificaciones de configuración que hayamos realizado.

Carpeta Imágenes

En ella estarán algunas de las imágenes que emplearemos en nuestra aplicación, como por ejemplo los gifs de carga de los controles UpdateProgress, o las imágenes necesarias para los Foros.

Carpeta Login



En esta carpeta ubicaremos todos los controles y páginas que tiene que ver con la gestión del registro y acceso de los usuarios al portal. Es el caso de, por ejemplo, la página de registro, o la de recuperación de contraseña. También incluiremos aquí los archivos de texto referentes a los e-mails enviados por el sistema en tareas relacionadas con el registro de usuarios.

Finalmente, y fuera ya del ámbito de cualquier carpeta, nos encontraremos con los ficheros correspondientes a la página maestra, la página por defecto, y el archivo de configuración web.config.