@dministración electrónica

Observatorio de Accesibilidad

GESTIÓN DE LA ACCESIBILIDAD EN GESTORES DE CONTENIDOS

GOBIERNO DE ESPAÑA
MINISTERIO DE HACIENDA
MINISTERIO DE POLÍTICA TERRITORIAL Y FUNCIÓN PÚBLICA

FICHA CATALOGRÁFICA

TÍTULO: Gestión de la accesibilidad en gestores de contenidos

Elaboración y coordinación de contenidos:

Secretaría General de Administración Digital (SGAD)

2ª edición electrónica: febrero de 2019

Disponible esta publicación en el Portal de Administración Electrónica (PAe): http://administracionelectronica.gob.es/

Edita:
© Ministerio de Política Territorial y Función Pública
Secretaría General Técnica
Subdirección General de Recursos, Publicaciones y Documentación

© Ministerio de Hacienda
Secretaría General Técnica
Subdirección General de Información, Documentación y Publicaciones
Centro de Publicaciones

Colección: administración electrónica

NIPO MPTFP: 277-19-043-2
NIPO MINHAC: 185-19-103-2

ÍNDICE

ÍNDICE DE IMÁGENES

1. INTRODUCCIÓN

Un **CMS** (*Content Management System*) o **sistema de gestión de contenidos** es un software alojado en un servidor Web que se apoya en una o varias bases de datos controladas a través de un interfaz en cliente, que se utiliza principalmente para facilitar la **creación**, **gestión**, **publicación** y **presentación** de grandes sitios Web, ya sea un sitio Web en Internet o una intranet corporativa.

Es muy importante tener presente que todo contenido Web debe ser creado y gestionado de forma que no suponga una barrera en cuanto a su acceso y empleo. Es aquí donde entra en juego el concepto de **Accesibilidad**, que puede definirse como la posibilidad de que **un sitio o servicio Web** pueda ser visitado y utilizado de forma satisfactoria por el mayor número posible de personas, independientemente de las limitaciones personales que tengan o de aquellas limitaciones que sean derivadas de su entorno.

Así, la **Accesibilidad Web** es un elemento esencial que favorece la igualdad de oportunidades de las personas con discapacidad, permitiendo el ejercicio del derecho reconocido constitucionalmente como es el acceso a la cultura, el ocio y el tiempo libre.

1.1. ¿CUÁLES SON LAS PRINCIPALES CARACTERÍSTICAS DE UN GESTOR DE CONTENIDOS?

Por lo general, un CMS proporciona un **editor de texto** WYSIWYG (*What You See Is What You Get*), mediante el cual el usuario puede visualizar el resultado final mientras escribe, similar a los convencionales procesadores de texto, pero con un rango de formatos de texto limitado.

Los CMS cuentan además con un conjunto de herramientas para definir la **estructura**, el **formato** de las páginas, el **aspecto visual**, uso de patrones, así como un **sistema modular** que permite incluir funciones no previstas inicialmente.

Una vez creados los documentos, se almacenan en una **base de datos central**, junto con los datos relativos a los mismos (versiones creadas, autor, fecha de publicación y caducidad, etc.), datos y preferencias de los usuarios, la estructura de la Web, etc.

La **determinación de perfiles** es imprescindible para facilitar el ciclo de trabajo, con un circuito de edición que va desde el autor hasta el responsable final de la publicación. El CMS permite la comunicación entre los miembros del grupo y hace un seguimiento del estado de cada paso del ciclo de trabajo.

Con el fin de obtener un sitio Web con un aspecto consistente, cuando se publica una página, se aplica el patrón definido para todo el sitio o para la sección concreta donde está situada. Esta **separación entre contenido y presentación** permite que se pueda modificar el aspecto visual de un sitio Web sin afectar a los documentos ya creados y por tanto, los autores de contenido no tienen que preocuparse por el diseño final de sus páginas.

Un CMS puede **gestionar automáticamente la accesibilidad de la Web**, con soporte de normas internacionales de accesibilidad como WAI, y adaptarse a las preferencias o necesidades de cada usuario. También puede proporcionar compatibilidad con los diferentes navegadores disponibles en todas las plataformas (Windows, Linux, Mac, Palm, etc.) y su capacidad de internacionalización le permite adaptarse al idioma, sistema de medidas y cultura del usuario.

1.2. APLICACIONES DE LOS GESTORES DE CONTENIDO

Los gestores de contenido manejan diferentes tipos de información, en función de los cuales se establecen sus aplicaciones:

- **Portales**: Sitios que combinan varias características (blogs, foros, noticias, buscadores, etc.) para crear una comunidad en línea.
- **Sitios empresariales o personales**: Sitios que proporcionan información de interés sobre una empresa o persona.
- **Blogs**: Generalmente son espacios personales en los que se publican cronológicamente artículos o noticias que pueden ser comentados pero no editados por los visitantes del sitio.
- **Foros**: Foros de discusión o debate en línea donde los usuarios opinan sobre temas de interés común.
- **Wikis**: Sitios con páginas en las que los usuarios aportan sus conocimientos mediante la escritura de artículos sobre algún tema de forma colectiva, pudiéndose crear, modificar o borrar un mismo texto compartido.

1.3. VENTAJAS DE LOS GESTORES DE CONTENIDO

El empleo de gestores de contenido conlleva una serie de ventajas, entre las que se encuentran las siguientes:

- Separación de contenido y presentación, lo que facilita los cambios de diseño.
- Creación, modificación y publicación de páginas Web más sencilla para un mayor número de usuarios.

- Incorporación, en el caso de los gestores de contenido más conocidos, de editores de texto visuales WYSIWYG que facilitan la labor de edición.
- División en módulos que facilita la incorporación de nuevas funcionalidades.
- Gestión dinámica de usuarios y privilegios mediante la posibilidad de establecer perfiles de usuario.
- Contenidos almacenados en base de datos, lo que facilita la exportación, catalogación, búsqueda y reutilización de contenidos.
- Gestión de los metadatos de cada documento, versiones, publicación y caducidad de páginas y enlaces rotos.

1.4. TIPOS DE GESTORES DE CONTENIDO

Los gestores de contenido se suelen clasificar en base a tres aspectos: **lenguaje de programación**, **tipo de licencia** y **funcionalidad**. Existen numerosos gestores de contenidos en el mercado y por lo tanto resulta imposible recoger un listado detallado de todos los posibles. En los siguientes apartados se recogerán algunos ejemplos de productos pero no son todos los existentes y el orden en el que aparecen no sigue ningún criterio de selección.

1.4.1. Por el lenguaje de programación empleado

Existen gestores de contenido basados en diferentes lenguajes de programación, como son:

- **Java**: jAPS, Liferay, DSpace, Fedora, Nuxeo EP, Magnolia, Hippo CMS, Calenco, Polopoly, IBM Lotus Web Content Management, Day Communiqué WCM, Jarimba, Vignette…
- **PHP**: Drupal, CMS Made Simple, Joomla!, Mambo, PHP-Nuke, TikiWiki, TYPO3, WordPress, Xoops, Zikula, Jadu, ExpressionEngine, Accrisoft Freedom, CMS 10, Dim Works CMS, Content-SORT, Prodigia Easy Site Manager, PipePS, SiteAd CMS…
- **ASP.NET**: DotNetNuke Community Edition, Umbraco, mojoPortal, Kentico CMS, SharePoint Server, Telligent Community, Ektron CMS400.NET, Quantum Art QP7, webControl CMS…
- **Otros (Perl, Python, Ruby,...)**: Blosxom, Bricolage, MojoMojo, Movable Type, TWiki, Scoop, Slash, Web GUI, Django-cms, MoinMoin, Plone, MediaCore, Radiant, Typo, Voranet CMS, VRContents...

1.4.2. Por el tipo de licencia

Atendiendo al tipo de licencia, los gestores de contenido pueden ser:

- **De código abierto** (*Open Source*): Generalmente, no tienen coste de licencia y su código puede ser modificado por cualquier desarrollador. El soporte de este tipo de CMS se basa en comunidades online de usuarios y su documentación en ocasiones puede ser escasa. Dentro de esta categoría, algunos de los gestores más utilizados son *WordPress*, *Drupal*, *Joomla*, *Plone*, *TYPO3*, *OpenCMS*, *PHPNuke* o *Moodle.*
- **De código propietario**: Tienen coste de licencia y su código sólo puede ser modificado por su desarrollador. No obstante, ofrecen un soporte profesional estable, así como una rica información documental. En esta categoría se encuentran gestores como *CMS10*, *Eximius2 CMS*, *Contendo* CMS, *Jarimba*, *CMS HYDRAportal*, *OnBase*, *IWEB*, *Oracle Portal*, *PipePS*, *Paloo*, *Smartone CMS*, *Vbulletin*, *XCM – Xeridia Content Manager*, *ZWeb Publisher CMS*…

1.4.3. Por tipo de uso o funcionalidad

- **Plataformas generales**: Drupal, Gekko, E107, Joomla, Mambo, PHP-Nuke, TYPO3, TYPOLight, XOOPS, ZWeb Publisher CMS, ADSM Portal 2.0, 360 Web Manager Software, GTLive! …
- **Sitios educativos**: ATutor, Claroline, Dokeos, eCollege, FrogTeacher, Moodle, Sakai Project, Scholar360, Synergeia, Teletop…
- **Blogs**: WordPress, bBlog, DotClear, Lifetype, Plone, Nucleus CMS, Blogger, Textpattern…
- **Galerías**: Gallery, Pixelpost, Expression Engine…
- **Wikis**: MediaWiki, TikiWiki, TWiki…
- **Comercio electrónico**: osCommerce, Magento, Zen Cart, Drupal e-Commerce, CubeCart, Open cart, VirtueMart…
- **Groupware**: Webcollab, eGroupWare, Groupware…

2. OBJETIVO DE LA GUÍA

El objeto de la presente guía es el de ofrecer una visión general de la gestión de la **accesibilidad propia de los gestores de contenido**. Pretende ser una ayuda para técnicos administradores de sitios Web, en la cual se recojan las cuestiones de accesibilidad que deben tener en cuenta e implementar en un gestor de contenidos para conseguir sitios accesibles.

3. FASES DE UN PROYECTO WEB SOBRE UN SISTEMA CMS

Un desarrollo Web realizado sobre un sistema de gestión de contenidos se suele llevar a cabo en una serie de fases más o menos complejas, pudiéndose definir como tarea transversal lo que se conoce como la **gestión de la accesibilidad del gestor de contenidos**. Esta tarea se aplica a cualquier proyecto implementado sobre el sistema de gestión de contenidos y consiste en solucionar los problemas de accesibilidad que éste incorpora por defecto.

3.1. DEFINICIÓN INICIAL DE LA ESTRUCTURA GENERAL DEL SITIO WEB

Se trata de definir con el cliente entre otras cosas, la **estructura del sitio** (teniendo en cuenta criterios de accesibilidad como la coherencia de navegación, la claridad del lenguaje o las tecnologías usadas), la **integración con otros sistemas** o la posibilidad de contar con **funcionalidades añadidas** a las propias del gestor de contenidos.

3.2. CREACIÓN DE UN DISEÑO GRÁFICO Y ADAPTACIÓN ENTRE LA ESTRUCTURA DEFINIDA Y EL DISEÑO

Se desarrolla el **diseño gráfico**, que debe ser adaptado lo máximo posible a la arquitectura de información definida en la fase anterior. Es el momento de hacer especial énfasis en la accesibilidad del sitio, estableciendo un diseño coherente, mecanismos de navegación claros, contrastes adecuados y elementos de interfaz fácilmente identificables. Se evitará el empleo de elementos que puedan constituir un obstáculo a la accesibilidad (por ejemplo, un diseño centrado exclusivamente en Flash). En caso de que el diseño requiera el uso de JavaScript se ha de tener en cuenta que esta tecnología debe usarse de forma compatible con la accesibilidad. Es decir, los efectos y funcionalidades basados en JavaScript deben ser compatibles con los lectores de pantalla y otros productos de apoyo. En caso de que esto no sea así entonces no se puede depender del soporte de JavaScript y el sitio web deberá ser funcional cuando JavaScript no se ejecute.

3.3. IDENTIFICACIÓN DE LOS DIFERENTES MODELOS DE PÁGINA DETECTADOS

Se definen los **modelos de página** que el gestor de contenidos ofrecerá al usuario, identificando aquellas zonas que serán editables por el editor final de contenidos, y aquellas que se deben mantener automáticamente por el sistema.

3.4. MAQUETACIÓN ESTÁTICA DE LOS MODELOS DE PÁGINA

Se realiza una **maquetación estática** de cada modelo de página detectado, aplicando criterios de accesibilidad, que se mantendrán a lo largo de todo el desarrollo.

3.5. DESARROLLO DE LA PLANTILLA BASE

Se trata de una fase que no es viable en todos los gestores de contenido. Consiste en desarrollar en un único punto la **estructura básica** y las **funcionalidades comunes** de la página, respetando en todo momento la premisa de separar adecuadamente contenido de presentación, lo cual permitirá reducir errores. Cualquier corrección que se realice se propagará a todas las plantillas del proyecto.

3.6. DESARROLLO DE LOS ELEMENTOS COMUNES

Los **elementos comunes** de las páginas (cabecera, pie, banners, formulario de búsqueda, menús de navegación, etc.) son **generados de forma automática** por el sistema, lo que evitará en gran medida la posibilidad de que el usuario editor incurra en disconformidades de accesibilidad. Es necesario que cuando se desarrollen estos elementos el código fuente sea correcto, tanto gramatical como semánticamente.

3.7. DESARROLLO DE LAS PLANTILLAS PARA LOS DIFERENTES MODELOS DE PÁGINA

Se desarrollan las **plantillas** a las que accederán los usuarios finales para la gestión de los contenidos, manteniendo siempre la estructura definida en la maquetación e intentando atomizar al máximo la información de la página. Por ejemplo, es preferible una plantilla que cuente con múltiples campos y de formato automáticamente (por ejemplo campos para el título, entradilla, cuerpo, archivos adjuntos, etc.), que una plantilla que permita incluir texto con formato en un único campo y sea el editor final el que cree la presentación de la página.

3.8. PRUEBAS Y CORRECCIONES

Se trata de una fase fundamental en cualquier proyecto Web, cobrando especial relevancia la **accesibilidad**. Así, resulta aconsejable revisar y corregir aquellas disconformidades que puedan comprometer el correcto acceso de los usuarios a los contenidos y funcionalidades de las páginas.

4. ASPECTOS DE ACCESIBILIDAD QUE SE DEBEN TENER EN CUENTA EN LA IMPLANTACIÓN DE UN GESTOR DE CONTENIDOS

La **accesibilidad** de un sitio Web depende de tres factores: la accesibilidad propia del gestor de contenidos, el desarrollo de las plantillas y la gestión de contenidos.

La clave principal para mantener la accesibilidad de los contenidos radica en la **accesibilidad propia del gestor de contenidos**, es decir, en las opciones de que disponga para crear contenidos accesibles: proporcionar textos alternativos, crear listas, tablas, encabezados, etc. Si el gestor de contenidos no posibilita estas opciones será imposible hacer el sitio web accesible.

Pero también es importante vigilar la implantación concreta del gestor. Puede suceder que de dos sitios web implementados con el mismo gestor de contenidos uno de ellos sea accesible y el otro no, al tener deficiencias en la implementación de las plantillas o en los diseños elegidos, etc.

Por último, es crucial el papel que juegan los editores finales de contenidos que son los responsables diarios de la introducción de nuevos contenidos. Así, el diseño de un sitio Web puede ser totalmente accesible y dejar de serlo en el momento en el que se incluyen nuevos contenidos. El gestor de contenidos y su implantación deben intentar hacer que la operativa de estos usuarios, manteniendo la accesibilidad, sea lo más sencilla posible. Adicionalmente a esto, es imprescindible que los editores finales de contenidos estén correctamente formados para generar contenidos accesibles puesto que hay cuestiones que sólo ellos mismos podrán introducir y controlar. Para profundizar en las tareas que deben realizar los editores finales de contenidos puede consultar la guía elaborada también por el Observatorio de Accesibilidad: "**Guía de cuestiones básicas de accesibilidad para editores finales de contenidos**", disponible en el área de documentación del Portal de la Administración Electrónica (PAe)[1].

En los siguientes apartados se explicarán las cuestiones de accesibilidad que deben tener en cuenta los técnicos administradores al implantar un gestor de contenidos. Se estudiarán las opciones de accesibilidad que un gestor de contenidos ha de reunir para generar contenidos finales accesibles.

[1] Área de documentación del Portal de la Administración Electrónica: http://administracionelectronica.gob.es/PAe/accesibilidad/documentacion

4.1. INCLUSIÓN DE IMÁGENES

Toda imagen incluida en una página Web, ya sea informativa, funcional, textual, decorativa o compleja, debe proporcionar un texto alternativo que aporte la misma información o función que la imagen.

El texto alternativo es la información proporcionada por los agentes de usuario (navegadores y productos de apoyo) en caso de no mostrarse la imagen. Por ejemplo, los navegadores gráficos que tengan las imágenes desactivadas o no puedan mostrarlas, y los navegadores de texto, usarán el texto alternativo en su lugar. Por otra parte, los lectores de pantalla leerán en voz alta el texto alternativo de las imágenes.

Toda imagen, cuyo contenido muestre información necesaria para comprender el contenido de la página, debe tener definido su texto alternativo. Este texto alternativo deberá ser:

- Descriptivo del contenido de la imagen.
- No demasiado largo.
- Preferentemente sin abreviaturas.

Aquellas imágenes simplemente decorativas deberán tener un texto alternativo vacío (*alt*="").

A la hora de insertar una imagen mediante un gestor de contenidos, entre las propiedades de la misma debe existir un campo para introducir su **texto alternativo**. Dicho texto será el contenido del atributo *alt* que se genere para la imagen en el código fuente de la página.

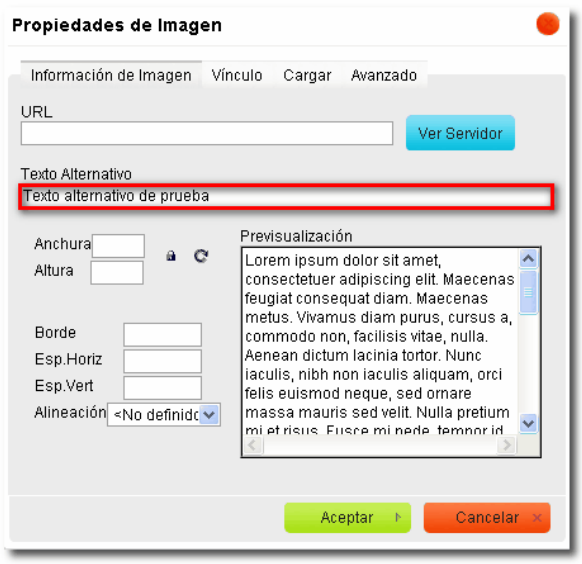

Figura 1. Campo de introducción de texto alternativo para una imagen

Ejemplo de código 1

```
<img alt="Texto alternativo de prueba" src="../Prueba.jpg" />
```

Se deberá vigilar que en el caso de dejar este campo en blanco, al tratarse de una imagen decorativa, se genere el necesario atributo *alt* vacío.

Ejemplo de código 2

```
<img alt="" src="../Prueba.jpg" />
```

Cuando la imagen sea decorativa, además de dejar el texto alternativo vacío, es importante que la herramienta no asigne un título a la imagen (atributo *title*). El objetivo es que los lectores de pantalla no lean textos alternativos superfluos que puedan entorpecer el uso de la página. Existen lectores de pantalla que si no encuentran un texto alternativo entonces leen el título. Para evitar eso, también es necesario que las imágenes decorativas carezcan de un título.

En ocasiones, el editor final necesita introducir imágenes complejas a través de las cuales se transmite mucha información (gráficas, diagramas, mapas, etc.), y que por lo tanto no puede ser descrita en pocas palabras. En estos casos, además de ofrecer una alternativa textual que identifique brevemente el tipo de información transmitida por la imagen, se debe proporcionar una descripción detallada en una página aparte o en la misma página en la que se encuentra la imagen.

En (X)HTML el elemento `IMG` dispone del atributo *longdesc* donde se debe indicar la URL de la ubicación donde se encuentra la descripción larga de la imagen.

Adicionalmente, y para ofrecer la máxima compatibilidad, se puede proporcionar un enlace de texto a continuación de la imagen, vinculándolo al comienzo de la descripción detallada.

Por lo tanto, el gestor de contenidos deberá incluir un campo, entre las propiedades de la imagen, para indicar la **URL de la descripción larga** de una imagen compleja.

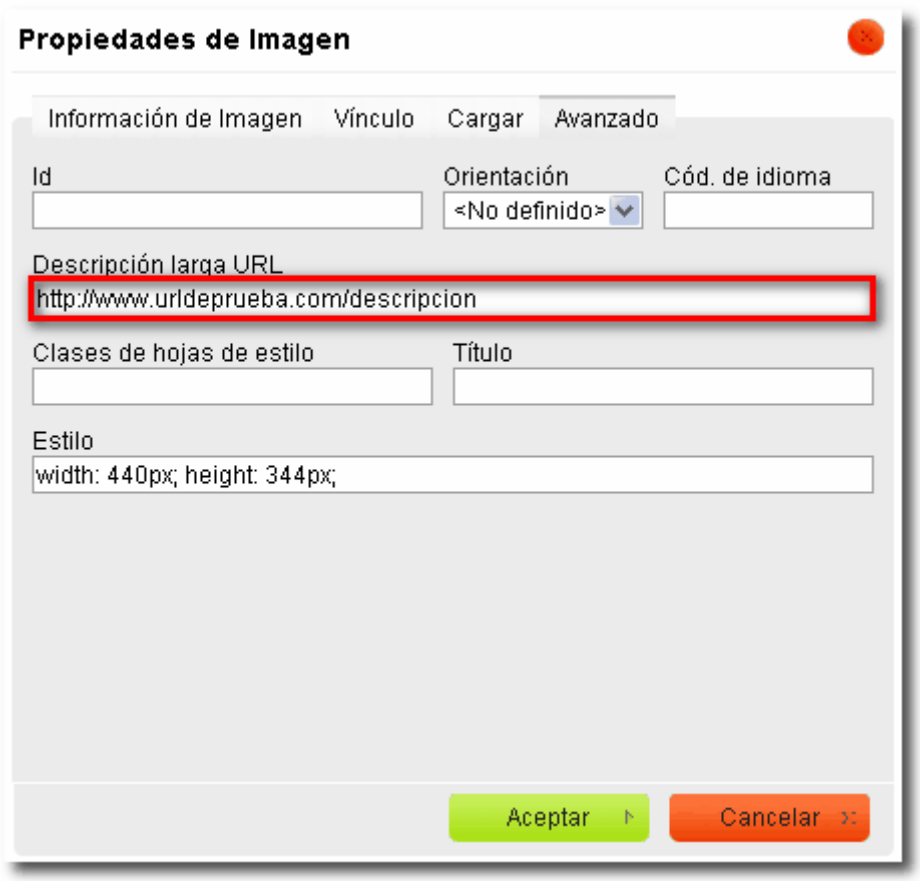

Figura 2. Campo de introducción de URL de descripción detallada para imágenes complejas

Ejemplo de código 3

```
<img alt="Texto alternativo de prueba" long-
desc="http://www.urldeprueba.com/descripcion" src="../Prueba.jpg"
/>
```

Sin embargo, en HTML5 el atributo *longdesc* se considera obsoleto (*deprecated*) debido al escaso soporte que le han dado los navegadores.

Por tanto, en caso de que la gramática empleada en el sitio web sea HTML5 no es necesario que el gestor de contenidos proporcione a los editores la posibilidad de definir la URL para el atributo *longdesc*. En este caso, la forma de incluir una descripción detallada es incluyendo dicha descripción en la misma página o en una página aparte enlazada con un enlace situado de forma contigua a la imagen. Este requisito pasa por tanto a ser una responsabilidad exclusiva del editor de contenidos.

También es aconsejable que el gestor de contenidos incluya una funcionalidad que permita indicar las **medidas finales de la imagen**, tanto el ancho como el alto. De este modo, al cargarse el documento, el navegador tendrá en cuenta el tamaño que va a ocupar la imagen y no se producirán ajustes continuos en la maquetación de los contenidos a medida que se cargan las imágenes.

Figura 3. Campo de introducción de Anchura y Altura de la imagen

4.2. DEFINICIÓN DE ENCABEZADOS O TÍTULOS DE PÁGINA

Las páginas con un volumen de información elevado suelen dividirse en una serie de capítulos, apartados, secciones y párrafos. Estos trozos semánticos de información constituyen la estructura de la página que es usada para navegar por los usuarios invidentes (y con otras discapacidades). De este modo, pueden acceder de forma más sencilla a la información que necesitan sin tener que esperar a que les lean la página completa. Es por ello que los encabezados de página se consideran elementos básicos en la navegación de un sitio web.

En (X)HTML, la estructura de una página se define a través de elementos de encabezado o título, con diferentes niveles de profundidad, permitiendo acceder rápidamente a las diferentes secciones de ésta.

Los encabezados deben seguir una estructura jerárquica, según el nivel de profundidad sin que se produzcan saltos en los niveles identificados y sin incluir encabezados vacíos.

Es necesario que los gestores de contenido ofrezcan opciones para marcar los diferentes **niveles de encabezado o título** existentes en las páginas de un sitio web. Es muy importante que cuando se identifique un encabezado, el propio gestor de contenidos realice la transformación por medio de los elementos HTML adecuados (`H1-H6`), y no mediante características de presentación (por ejemplo, efectos de fuente).

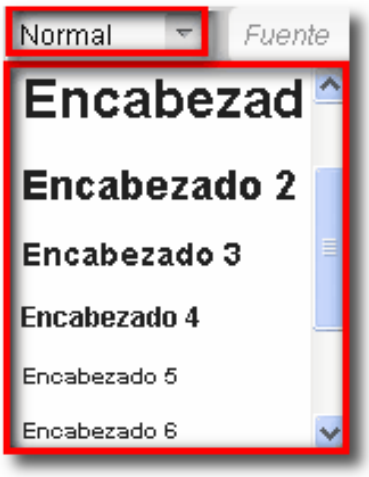

Figura 4. Funcionalidad de selección de niveles de encabezado para una página

Ejemplo de código 4

```
<h1>Noticias</h1>
<p>Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed
do eiusmod tempor incididunt ut labore et dolore magna aliqua.</p>
<h2>Noticias de Junio 2010</h2>
<ul>
<li>Lorem ipsum dolor sit amet</li>
<li>Lorem ipsum dolor sit amet</li>
<li>Lorem ipsum dolor sit amet</li>
</ul>
<h2>Noticias de Julio 2010</h2>
<ul>
<li>Lorem ipsum dolor sit amet</li>
<li>Lorem ipsum dolor sit amet</li>
<li>Lorem ipsum dolor sit amet</li>
</ul>
```

Como los encabezados de una página deben seguir el nivel jerárquico, sin saltarse niveles intermedios, el primer encabezado del contenido editado en el gestor de contenidos debe ser del nivel adecuado para que la estructura de encabezados de la página, una vez que se haya añadido dicho contenido, sea la correcta.

Para ello esta responsabilidad se puede dejar en manos de los editores finales de contenido a los que se les instruye para comenzar cada contenido en un nivel determinado. Otra opción es dejar que se edite el contenido comenzando con un encabezado `H1` y que sea la herramienta la que haga la conversión de forma automática a los niveles adecuados. Finalmente, otra opción es eliminar en el gestor de contenidos los niveles que no se pueden usar. Por ejemplo, si el contenido debe empezar en un `H3` se pueden eliminar las opciones para crear encabezados de nivel `H1` y `H2`.

En el caso de que la gramática empleada sea HTML5 este problema se ve mitigado ya que es posible incluir el contenido editado dentro de un elemento de sección como `ARTICLE` o `SECTION`. Como en HTML5 cada elemento de sección puede comenzar por un `H1`, independiente de la estructura anterior de encabezados, entonces los gestores y editores finales de contenidos no tendrán que preocuparse por el nivel inicial de encabezados, comenzando siempre en un `H1`.

Ejemplo de código 5

```
<h1>Título de la página</h1>
<p>Lorem ipsum dolor sit amet</p>
<article>
<h1>Título del contenido editado</h1>
<p>Lorem ipsum dolor sit amet</p>
<p>Lorem ipsum dolor sit amet</p>
</article>
```

La estructura u *outline* del ejemplo anterior en HTML5 respeta la jerarquía del documento:

- Título de la página
    - Título del contenido editado

4.3. INCLUSIÓN DE ENLACES

Los usuarios invidentes y con otras discapacidades se ayudan en la navegación de herramientas que les permiten mostrar un listado de todos los enlaces de una página. Por lo tanto, la descripción del enlace debe ser lo suficientemente buena para comprender su utilidad fuera del contexto. Por ejemplo deben evitarse textos para los enlaces como “pinche aquí”, “más”, etc.

Excepcionalmente, cuando el texto del enlace no es descriptivo por sí mismo, al menos debe serlo mediante su contexto más inmediato. Se entiende por contexto inmediato aquellos elementos que un lector de pantalla puede obtener a partir del enlace como el texto de la frase, el párrafo, la celda de tabla o el elemento de lista que contiene al enlace, o bien el encabezado de la sección en la que se encuentra.

Por otro lado, se ha de evitar la **apertura de enlaces en nuevas ventanas** del navegador dados los problemas que ello puede originar:

- El usuario puede desorientarse, al no darse cuenta de lo que ha ocurrido.
- La nueva ventana tendrá anulada la funcionalidad del botón "*atrás*".
- El rendimiento del sistema puede verse reducido.
- Puede confundir al usuario, en caso de que no entienda que la nueva ventana es realmente una ventana del mismo navegador que estaba usando.
- El usuario se puede sentir confuso, puesto que los navegadores modernos bloquean en ocasiones la apertura de nuevas ventanas, lo que le puede hacer pensar que el enlace no funciona.

En caso de que dicha apertura resulte completamente necesaria, se recomienda informar de la misma. Concretamente, para las Administraciones Públicas, se

considera como necesaria o recomendable la apertura de ventana nueva en los siguientes casos:

- Enlaces a sitios web externos.
- Enlaces a archivos adjuntos.

En las WCAG 2.1 la obligatoriedad de avisar de la apertura de nuevas ventanas o pestañas del navegador pasa a ser un requisito de AAA, aunque sigue siendo una buena práctica y es muy recomendable su uso ya que mejora tanto la accesibilidad como la usabilidad de la página. En principio es suficiente con avisar por medio del atributo *title* del enlace.

Pese a que la técnica anterior es suficiente, a continuación se exponen los métodos más aconsejables para avisar de la apertura de enlaces en nueva ventana.

Para los **enlaces de texto**:

- En el texto del propio enlace:

Ejemplo de código 6

```
<a href="nueva_ventana.html" target="_blank">Texto del vínculo
(se abre en ventana nueva)</a>
```

- Aportando un elemento gráfico () que indique al usuario visualmente (y a través de su alternativa) la apertura de nueva ventana:

Ejemplo de código 7

```
<a href="nueva_ventana.html" target="_blank">Texto del vínculo
<img alt="Se abre en ventana nueva" src="icono.gif" /></a>
```

Para los **enlaces gráficos**:

- Como texto del enlace, o bien, aportando un elemento gráfico () que indique al usuario visualmente (y a través de su alternativa) la apertura de nueva ventana.
- Aportando la información de apertura en nueva ventana en el contenido de la imagen. Para ello se puede incluir el elemento gráfico de apertura en nueva ventana () dentro de la propia imagen, indicando al usuario visualmente y a través de su alternativa la apertura de nueva ventana:

Ejemplo de código 8

```
<a href="nueva_ventana.html" target="_blank">
<img src="imagen_con_icono_de_apertura.jpg"
alt="alternativa_textual_imagen. Se abre en ventana nueva"/></a>
```

Una solución válida tanto para enlaces textuales como para enlaces gráficos consiste en incluir el texto "*Se abre en nueva ventana*" en el propio enlace, mostrándolo a modo de *tooltip* mediante técnicas CSS cuando se fija el foco sobre el enlace.

Los gestores de contenido deberían incluir entre las propiedades de enlace un campo que permita indicar **información adicional** acerca del mismo o su posible **apertura en nueva ventana** mediante la inclusión de textos como: "*se abre en ventana nueva*". A nivel de código, el gestor incluirá esta información a través del atributo *`title`* del enlace.

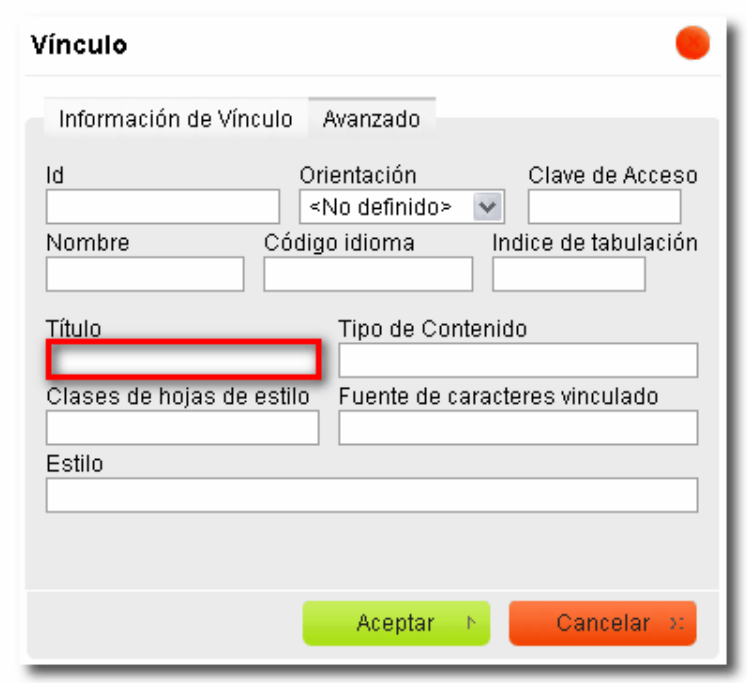

Figura 5. Campo de introducción de información adicional en enlaces

Lo óptimo sería que el propio CMS gestione directamente el aviso de apertura de nueva ventana. Por ejemplo, cuando se incluya un enlace que se abre en nueva ventana, el gestor de contenidos podría incluir automáticamente un icono de nueva ventana con su alternativa textual. De este modo se gana consistencia en la navegación, al avisar siempre de la misma forma.

4.4. INCLUSIÓN DE FICHEROS ADJUNTOS

Los ficheros adjuntos incluidos en los contenidos generados deberán disponer de un **texto de enlace descriptivo** del fichero que se vincula. Este texto descriptivo será normalmente el nombre o contenido principal del fichero adjunto y debe ser comprensible fuera de contexto.

Además de un texto descriptivo del fichero, es recomendable incluir en el título del enlace (atributo `title`) una indicación sobre el formato del mismo, junto con el texto replicado del enlace (por ejemplo: "*Accesibilidad en gestores de contenidos. Fichero PDF*"). El gestor de contenidos debe ofrecer un campo a través del cual se pueda indicar la información de formato del fichero adjunto.

Con el fin de diferenciar visualmente los enlaces a ficheros adjuntos, también es recomendable que el gestor disponga de una funcionalidad que permita aplicar a este tipo de enlaces algún estilo CSS o icono representativo del formato del fichero.

Adicionalmente a estas medidas, es importante recordar que los ficheros adjuntos deben ser también accesibles. Es decir, deberán adoptar las características de accesibilidad del formato usado para crear los documentos. Por ejemplo, si el adjunto es un documento PDF deberá ser un PDF etiquetado y accesible. En caso contrario, de no tratarse de documentos accesibles, entonces se ha de proporcionar al menos una alternativa en un formato que sí sea accesible (por ejemplo, en formato HTML).

Para ampliar información sobre la construcción de documentos PDF accesibles se puede consultar la ***"Guía de accesibilidad en PDFs"*** disponible en el área de documentación del Portal de la Administración Electrónica (PAe)[2].

4.5. IDENTIFICACIÓN DE LISTAS

Las listas permiten identificar grupos de elementos que tienen alguna relación entre sí, lo que ayuda a comprender la estructura de las páginas o de los contenidos. Los usuarios invidentes cuentan con herramientas que les permiten navegar por el contenido de las listas de una forma estructurada y más cómoda.

[2] Área de documentación del Portal de la Administración Electrónica: http://administracionelectronica.gob.es/PAe/accesibilidad/documentacion

En (X)HTML se distinguen tres tipos de lista:

- **Lista no ordenada**: conjunto de elementos relacionados entre sí para los que no se indica un orden o secuencia determinados. Este tipo de lista se define mediante el elemento <UL>, mientras que sus elementos se definen mediante la etiqueta <*li*>.
- **Lista ordenada**: conjunto de elementos relacionados que se muestran siguiendo un orden determinado. Este tipo de lista se define mediante el elemento <OL>, mientras que sus elementos se definen mediante la etiqueta <*li*>, la misma que se utiliza en las listas no ordenadas.
- **Lista de definición**: conjunto de elementos que están formados por términos y definiciones. La etiqueta <DL> crea la lista de definición y las etiquetas <*dt*> y <*dd*> definen respectivamente el término y la descripción de cada elemento de la lista.

El gestor de contenidos ha de ofrecer opciones para marcar **cualquier enumeración de elementos** como lista, ya sea no ordenada, ordenada o de definición. Es muy habitual que los editores visuales no incluyan una funcionalidad para crear listas de definición, por lo que en ese caso resulta fundamental que el equipo de desarrollo del sitio web la añada al editor. A nivel de código, la lista generada por el gestor deberá tener una estructura correcta y, en ningún caso, ser simulada mediante elementos que no han sido creados para tal fin (por ejemplo, párrafos iniciados con asterisco, guiones o números).

También se debe disponer de una funcionalidad que permita crear **listas anidadas**, es decir, definir listas completas en los elementos de otras listas, con independencia del tipo de lista del que se trate (por ejemplo una lista ordenada dentro de una lista no ordenada). Con ello se contribuye a aumentar el valor semántico del listado y se facilita la interpretación del mismo a determinados usuarios, por ejemplo, aquellos que hacen uso de lectores de pantalla.

Figura 6. Funcionalidades de marcado de listas ordenadas y no ordenadas

Figura 7. Funcionalidades de sangrado empleadas en el anidamiento de listas

4.6. INCLUSIÓN DE TABLAS DE DATOS

En (X)HTML, las tablas sirven para mostrar información tabular y no para dotar de presentación a los contenidos, por lo que se debe evitar el uso de tablas para maquetar. Así, las tablas de datos estructuran la información en filas y columnas describiendo una relación entre cada celda de datos con otras celdas en su misma fila y/o columna.

Uno de los requisitos principales de una tabla de datos es que cada celda de encabezado se identifique mediante el elemento `TH`. Con ello se consigue:

- Que los lectores de pantalla puedan asociar las celdas de datos con sus correspondientes encabezados.
- Que se indique visualmente los encabezados en navegadores gráficos.
- Que el editor final de contenidos pueda diferenciar el estilo con CSS.

Asimismo, en tablas de datos complejas (aquellas con dos o más niveles lógicos de encabezado) se debe realizar una asociación explícita entre las celdas de datos y las celdas de encabezado correspondiente, con el fin de permitir una correcta interpretación de la tabla por los productos de apoyo. Dicha asociación se realiza por medio de los atributos *`id`* y *`headers`*, de forma que cada celda de datos incluirá en su atributo *`headers`* el identificador unívoco *`id`* de todos los encabezados relacionados con ésta.

Cuando se crea una tabla con un gestor de contenidos, por defecto no se marcan los encabezados, si bien, éste debe ofrecer al menos herramientas de marcado semiautomático (marcado de determinadas filas y/o columnas de la tabla). Es muy importante que a la hora de identificar encabezados en tablas de datos, el propio gestor de contenidos realice la transformación por medio del marcado adecuado (`TH`), y no mediante características de presentación (por ejemplo, efectos de fuente y fondo).

También sería aconsejable que el gestor de contenidos ofreciera una funcionalidad para asociar celdas de datos y encabezados.

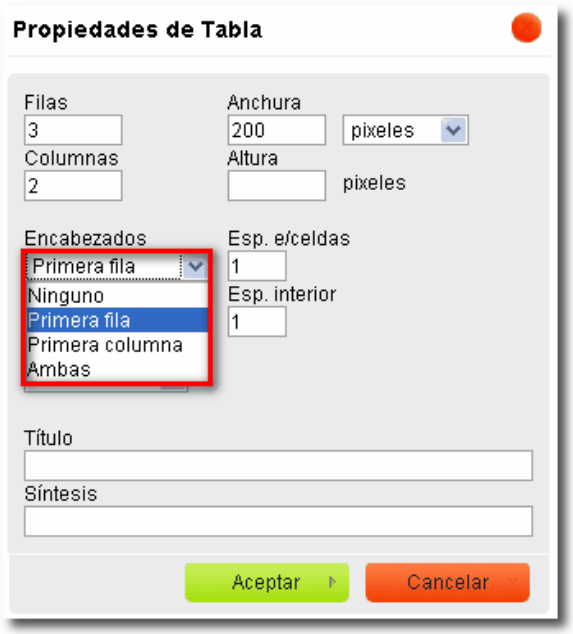

Figura 8. Funcionalidad de marcado de encabezados de tablas de datos

Por otro lado, es importante que el gestor de contenidos incluya entre las propiedades de la tabla unos campos para introducir un **título** que describa brevemente la naturaleza de la tabla, y/o un **resumen** de las relaciones entre las celdas de datos de la misma. En (X)HTML el título se proporciona a través del elemento `CAPTION`, mientras que el resumen se incluye por medio del atributo *`summary`*.

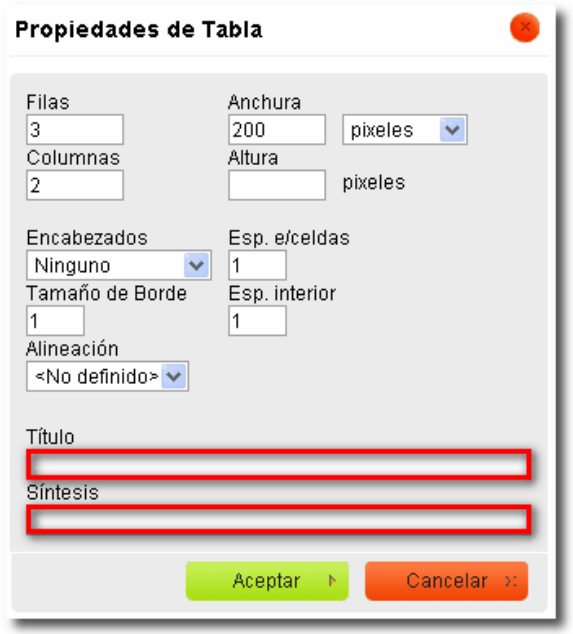

Figura 9. Campos de introducción de título y resumen de tablas de datos

Nota: En HTML5 el atributo *summary* de las tablas de datos se considera obsoleto. Sin embargo, en gramáticas anteriores sigue siendo válido y está contemplado como una técnica correcta para proporcionar un resumen para las tablas de datos. Por tanto, si la gramática empleada en el sitio web es HTML5 el gestor de contenidos no debería emplear el atributo *summary* para incluir información de resumen. Dicha información, si se le solicita al usuario, se puede incluir de forma alternativa como un párrafo previo a la tabla.

En ocasiones podría ser necesario definir un ancho específico para la tabla, por lo que sería recomendable disponer de una opción para ello.

Figura 10. Campo de selección del tipo de unidad para las dimensiones de una tabla

En caso de que se fijen propiedades visuales como el **borde**, el **espacio entre celdas** o el **espacio interior**, el editor visual debería introducirlas por CSS y no mediante atributos de presentación (`border`, `cellspacing`, `cellpadding`).

4.7. IDENTIFICACIÓN DE CAMBIOS DE IDIOMA

Identificar correctamente los cambios de idioma facilita la comprensión de los contenidos, entre otros, a los usuarios que utilizan lectores de pantalla o programas de síntesis de voz, debido a que éstos detectarán el cambio de idioma y verbalizarán correctamente el contenido.

Si se utilizan varios idiomas en una misma página, se debe asegurar que cualquier cambio de idioma esté indicado. Esta indicación varía en función de la gramática empleada:

- Páginas con gramática HTML: atributo *lang*.
- Páginas con gramática XHMTL 1.0 servido como text/html: atributos *lang* y *xml:lang*.
- Páginas con gramática XHTML 1.0 servido como xml: atributo *xml:lang*.
- Páginas con gramática XHTML 1.1: atributo *xml:lang*.

Los atributos de cambio de idioma se aplican sobre el elemento que contiene el texto en el idioma que cambia y pueden ser aplicados a cualquier elemento (X)HTML. En caso de que el cambio de idioma se encuentre en un fragmento de texto dentro de un párrafo, se puede marcar a través del elemento genérico SPAN.

Ejemplo de código 9

```
<p>En este caso usaremos preferentemente la modulación QPSK (<span
xml:lang="en" lang="en">Quadrature Phase Shift Keying</span>) en
lugar de 64-QAM (<span xml:lang="en" lang="en">Quadrature Ampli-
tude Modulation</span>).</p>
```

No será necesario identificar aquellos cambios de idioma derivados de palabras extranjeras que sean empleadas de forma común en la lengua de origen, ni de direcciones o de nombres propios.

Los gestores de contenido han de ofrecer una opción para poder asignar un idioma a un texto previamente seleccionado, para lo cual utilizarán a nivel de código el marcado (X)HTML indicado anteriormente.

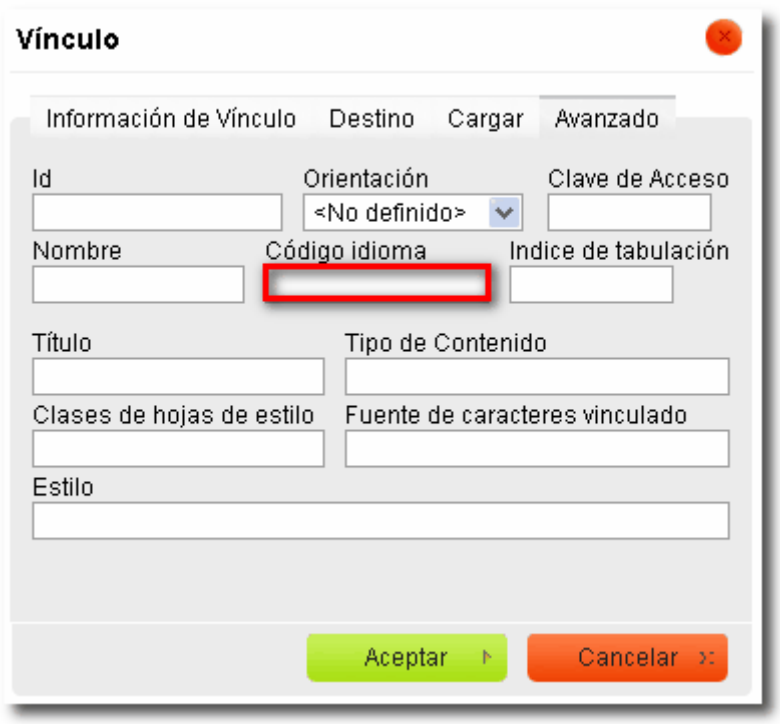

Figura 11. Campo de introducción del idioma de un elemento

4.8. IDENTIFICACIÓN DE CITAS

Una cita consiste en una referencia textual de un fragmento o totalidad del discurso de una persona, o el texto de otra fuente.

En (X)HTML existen dos tipos de cita:

- **Citas cortas o en línea**: fragmentos de texto contenidos en párrafos u otros elementos de bloque.
- **Citas largas o de bloque**: Uno o varios párrafos completos.

Figura 12. Funcionalidad de marcado de citas

Los gestores de contenido deben incluir una funcionalidad para marcar tanto citas en línea, como citas en bloque. A nivel de código, el propio gestor de contenidos deberá identificar las citas en línea mediante el elemento `Q` y las citas en bloque a través del elemento `BLOCKQUOTE`.

En el caso de las citas en bloque, es necesario que el texto interno del elemento `BLOCKQUOTE` se encuentre a su vez marcado con elementos de bloque (por

ejemplo como párrafo de texto). Además, es muy importante que el gestor no utilice este elemento para crear efectos visuales, tales como sangrías.

Ejemplo de código de cita en línea:

Ejemplo de código 10

```
<p>... Lorem ipsum dolor sit amet, consectetur adipisicing elit
<q>sed do eiusmod tempor incididunt ut labore et dolore magna ali-
qua</q></p>
```

Ejemplo de código de cita en bloque:

Ejemplo de código 11

```
<p>Lorem ipsum dolor sit amet, consectetur adipisicing elit:</p>
<blockquote><p>Lorem ipsum dolor sit amet, consectetur adipisicing
elit, sed do eiusmod tempor incididunt ut labore et dolore magna
aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco
laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure
dolor in reprehenderit in voluptate velit esse cillum dolore eu
fugiat nulla pariatur. Excepteur sint occaecat cupidatat non
proident, sunt in culpa qui officia deserunt mollit anim id est
laborum.</p></blockquote>
```

Otra funcionalidad interesante que podría incluir un gestor de contenidos es la de indicar la fuente de la que se ha extraído la cita. En (X)HTML es posible indicar la URL de la fuente de una cita mediante el atributo *cite*, así como marcar referencias mediante el elemento CITE.

4.9. USO DE UNIDADES RELATIVAS

Las **unidades relativas** (*em*, *ex* y *%*) especifican una medida en relación a otra propiedad de medida. Su empleo permite redimensionar el texto, lo que facilita el acceso a la información a usuarios con deficiencias visuales transitorias o permanentes y en general a todos los usuarios, de tal forma que puedan adaptar el tamaño de la fuente a sus preferencias o necesidades.

Por su parte, el uso de **unidades absolutas** (*px*, *pt*, *in*, *pc*, *cm* y *mm*) en fuentes, tablas o cualquier contenedor impedirá su redimensionado en ciertos navegadores, resultando útiles únicamente cuando las características físicas del medio de salida son conocidas.

Es muy común en los gestores de contenido la inclusión de funcionalidades de edición para aplicar tamaños de fuente a textos previamente seleccionados. Los textos editados a través de estas funcionalidades en principio han de quedar

definidos en **unidades relativas**, ya que de este modo podrán ser redimensionados por todos los usuarios según sus necesidades o preferencias.

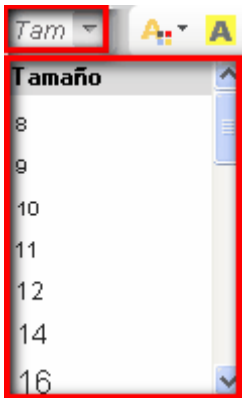

Figura 13. Funcionalidad de selección de tamaño de fuente para textos

Las WCAG 2.1 exigen que el texto se pueda redimensionar, sin necesidad de usar productos de apoyo específicos, al menos hasta el doble de su tamaño sin que se pierda contenido o funcionalidad. Sin embargo, el uso de unidades relativas para definir el tamaño del texto no es la única técnica posible para lograrlo.

Si todos los **navegadores de uso común** disponen de una función de **zoom** que permita aumentar el contenido manteniendo su legibilidad y funcionalidad entonces se considera que se cumple dicho requisito. Es decir, si los navegadores permiten hacer zoom no es necesario que el tamaño de texto se defina obligatoriamente en unidades relativas.

Sin embargo, para asegurar que se permite el redimensionado del texto en todos los navegadores, incluso en aquellos que carecen de zoom, es **recomendable** seguir empleando **unidades relativas** para definir el tamaño del texto y de sus contenedores.

4.10. MAQUETACIÓN ADAPTABLE

Relacionado con el apartado anterior, las WCAG 2.1 incluyen también como requisito que se pueda hacer zoom sobre al contenido hasta un 400% del tamaño original sin que se produzca un *scroll* en dos dimensiones. Es decir, si el *scroll* normal es en sentido vertical entonces no se puede producir un *scroll* horizontal al aumentar el zoom, y viceversa.

Permitir el reajuste del contenido se conoce también como Diseño Web Adaptable (*Responsive Web Design*) y en las Pautas de Accesibilidad WCAG 2.1 se considera como la forma más efectiva para conseguir cumplir este criterio. Así, entre las técnicas indicadas está el uso de ***media queries*** para establecer

puntos de ruptura y reformatear el contenido para diferentes anchos de visualización.

Por tanto, la maquetación de las plantillas empleadas en el gestor de contenido para las páginas de los sitios web ha de respetar este requisito. Deben poder adaptarse a diferentes tamaños de ventana o niveles de zoom sin que se pierda contenido o funcionalidad y sin que se produzca un doble *scroll*. Para ello se pueden emplear diferentes técnicas como la **maquetación fluida**, mediante **media queries**[3] y **grid layout**[4] de CSS o maquetación con el **modelo Flexible Box**[5] de CSS (Flexbox).

4.11. EDICIÓN DEL ESTILO O ASPECTO VISUAL DEL DOCUMENTO

Las funcionalidades ofrecidas por los gestores de contenido para cambiar aspectos presentacionales deben generar un código correcto, libre de elementos y atributos desaconsejados y/o de presentación. Las características desaconsejadas (elementos y atributos obsoletos) ponen en riesgo la compatibilidad con cualquier tipo de agente de usuario.

Así, por ejemplo, para enfatizar la información mediante negrita o cursiva se deben utilizar los elementos `STRONG` o `EM`, en lugar de los elementos desaconsejados `B` e `I`.

Del mismo modo, cuando se dota de estilo a un fragmento de texto (efectos comunes como el subrayado o tachado), el CMS siempre debe hacerlo mediante propiedades CSS y no con los elementos HTML desaconsejados (`U` y `STRIKE`).

Figura 14. Efectos de presentación a través de elementos desaconsejados

[3] Media queries: https://www.w3.org/TR/css3-mediaqueries/
[4] Grid layout: https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_Grid_Layout
[5] Flexible Box: https://www.w3.org/TR/2018/CR-css-flexbox-1-20181119/

Ejemplo de código incorrecto 1

```
<p><u>Prueba</u></p>
<p><strike>Prueba</strike></p>
```

Otro caso típico dentro de la edición de contenidos es la introducción de **saltos de línea** con el propósito de crear separaciones visuales en el contenido. Para conseguir tal efecto, el gestor de contenidos no debe generar **párrafos vacíos** con entidades * *.

Los efectos comunes en la presentación de contenidos en todas las páginas como son entre otros, el tipo de fuente, color del texto, espaciados o tabulaciones, se recomienda que el gestor de contenidos los lleve a cabo a través de **hojas de estilo CSS externas** y no mediante estilos en línea (atributo *style*) o embebidos (elemento STYLE), ya que éstos dificultan el mantenimiento y la limpieza del código.

Ejemplo de código incorrecto 2

```
<!-- Estilos CSS en línea -->
<img alt="" src="prueba.jpg" style="border-style: solid; border-
width: 5px; margin: 5px; width: 200px; height: 156px;" />
<p>Esto es un párrafo que incluye <span style="font-family: com-
ic sans ms,cursive;"> un fragmento de texto con una tipografía
diferente</span></p>
<p style="text-align: center;">Párrafo de texto centrado</p>

<!-- Estilos CSS embebidos -->
<style type="text/css">
  h1 {font-size: medium; border: solid; text-align: center}
</style>
```

Por lo tanto, sería recomendable eliminar de los editores visuales todas aquellas funcionalidades que generen características (X)HTML desaconsejados y/o de presentación en la aplicación de efectos visuales, o bien, mantenerlas siempre que éstas utilicen correctamente las propiedades CSS correspondientes.

También sería de gran utilidad que los gestores de contenido ofrecieran herramientas para pegar texto desde fuentes externas, de forma que se realice una transformación adecuada tanto a nivel visual como a nivel de código mediante el marcado HTML correspondiente: encabezados, listas, párrafos, etc.

4.12. MARCADO DE FORMULARIOS

Los formularios resultan esenciales en la experiencia de usuario de un sitio, al permitir interactuar con los servicios Web de empresas, instituciones, organismos

públicos, etc., dando acceso a servicios tales como: compras, banca, viajes, elecciones, y otros.

Un formulario se compone de dos tipos de elementos: **etiquetas** y **controles**.

Todo control de formulario, a excepción de los botones, debe ser identificado mediante una etiqueta, que debe ser marcada mediante el elemento `LABEL`. En caso de que en el diseño de la página no exista un texto visible que se pueda marcar como etiqueta con el elemento `LABEL`, entonces se puede usar el atributo *`title`* del campo de formulario para identificar cuál es su función.

Por otra parte, para garantizar la correcta interpretación de los formularios, es necesario que se lleve a cabo en ellos una asociación explícita entre etiquetas y controles.

La **asociación explícita** se realiza utilizando el elemento `LABEL` con el atributo *`for`*, de forma que el valor del atributo *`id`* de cada control coincida con el valor del atributo *`for`* de su respectiva etiqueta.

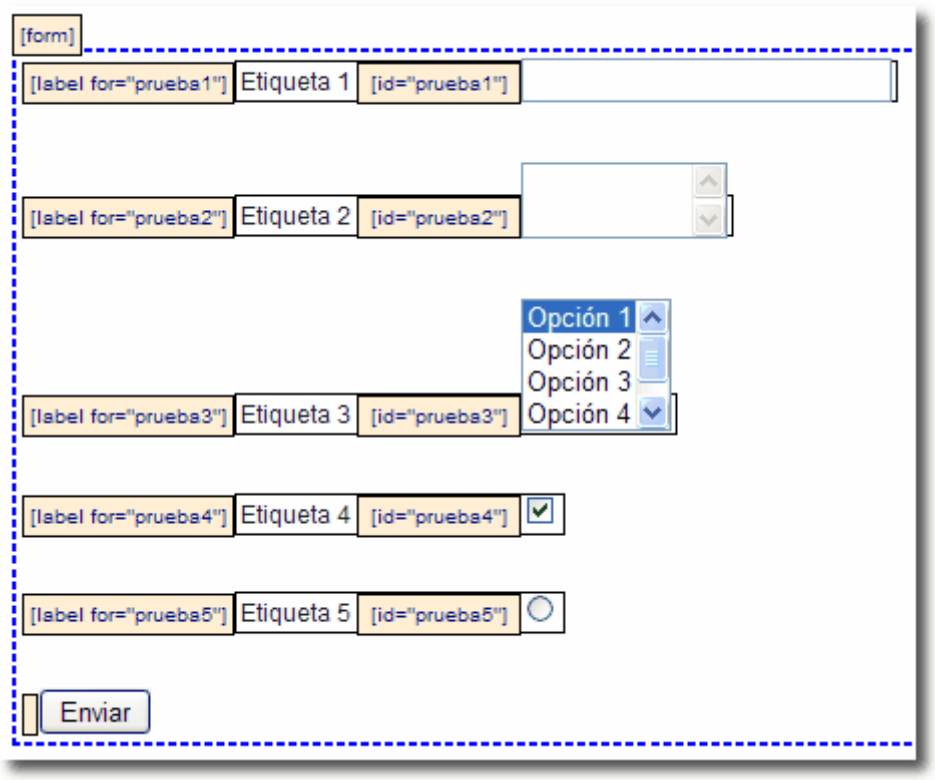

Figura 15. Ejemplo de formulario con asociación explícita

El hecho de que una etiqueta no esté identificada, implica que no sea posible asociarla a su correspondiente control explícitamente. Es por ello, que los gestores de contenido deben ofrecer una opción para identificar las etiquetas de

formulario, a través del marcado (X)HTML adecuado (elemento LABEL) y, por si no hubiese un texto que marcar como etiqueta, permitir proporcionar un título a los controles de formulario.

También se debe cuidar que los controles generados mediante el gestor dispongan de atributo *id*, ya que de no ser así, no podrán ser asociados de forma explícita con sus respectivas etiquetas.

Por ejemplo, cuando se crea un control de formulario mediante un gestor de contenidos, entre las propiedades del control, se podría incluir un campo para definir su correspondiente etiqueta, de forma que el gestor de contenidos genere automáticamente la etiqueta a nivel de código y asigne el mismo valor para el atributo *id* del control (INPUT, TEXTAREA o SELECT) y para el atributo *for* de la etiqueta (LABEL).

Ejemplo de código 12

```
<form action="prueba.html" enctype="text/plain" id="prueba" met-
hod="get" name="prueba">
<label for="nombre">Nombre: <input id="nombre" name="nombre" si-
ze="100" type="text" maxlength="100" /></label>
</form>
```

Respecto al texto escogido como etiqueta por los editores finales de contenido, en las WCAG 2.1 es necesario que éste coincida con el nombre accesible del campo de formulario (etiqueta LABEL o atributos *title*, *aria-label*, *aria-labelledby*, etc.). El objetivo de este requisito es que las personas que dependan de estas etiquetas visuales también puedan emplear los nombres accesibles. Así, una persona con una discapacidad motriz que pueda ver la página web en un navegador gráfico pero para interactuar emplee entrada por voz podrá leer el texto visible de la etiqueta para accionar o seleccionar el componente. Si la etiqueta visible y el nombre accesible no coinciden la interacción resulta mucho más complicada para este tipo de personas y se abre la posibilidad además de accionar componentes accidentalmente. Si el gestor de contenidos asigna las etiquetas a sus campos mediante asociación explicita, como se ha comentado antes, ya se está cumpliendo este requisito de coincidencia entre la etiqueta visible y el nombre accesible. El nombre accesible del campo será el texto de la etiqueta LABEL.

Adicionalmente, en las WCAG 2.1 es necesario identificar el propósito de los campos de introducción de datos de forma que se pueda determinar automáticamente la finalidad de aquellos que solicitan información acerca de las personas. De esta forma las aplicaciones podrán obtener esta información y facilitar la interacción a los usuarios. Por ejemplo, autocompletando los campos cuyos datos son conocidos.

Para ello el gestor de contenidos debería permitir que los editores finales de contenidos pudiesen escoger el tipo de información solicitada[6] en el campo y aplicar el marcado correspondiente. Esta selección se puede ofrecer mediante un menú desplegable donde los editores finales de contenidos puedan escoger entre algunos de los tipos de información personal más solicitada. Por ejemplo, *"Nombre completo"*, *"Título"*, *"Nombre propio"*, *"Apellidos"*, *"Dirección"*, etc.

Para aplicar el marcado semántico correspondiente, en algunos casos puede parecer suficiente con los nuevos tipos de elementos `INPUT` de HTML5 (tipo `tel`, `email`, etc.). Sin embargo, aunque estos campos aportan cierta información sobre la naturaleza del tipo de dato a introducir, las categorías que se pueden emplear son demasiado genéricas. Por ejemplo, sirve para indicar que un campo se trata de un email o un teléfono, pero no aclara cuál es el dato concreto al que se refiere (¿teléfono o email del usuario o de otra persona?).

Por lo tanto, es preciso emplear técnicas alternativas como el atributo `autocomplete` de HTML 5.2[7] junto con un valor que indique el tipo de información solicitada. Por ejemplo, algunos de los valores posibles para el atributo `autocomplete` son: `name` para indicar el nombre completo, `honorific-prefix` para el título *("Sr."*, *"Sra."*, *"Dr."*, etc.), `given-name` para el nombre propio, `family-name` para el apellido, `nickname` para el apodo, `street-address` para la dirección, etc.

Por otro lado, cuando en un formulario se utilizan varios controles relacionados entre sí (por ejemplo en un formulario de registro, los campos relativos a la información personal, los datos de contacto, las áreas de interés, etc.), es recomendable agruparlos para facilitar su comprensión. Para ello se utiliza el elemento `FIELDSET`, el cual debe contener a su vez un elemento `LEGEND` que proporcionará un título identificativo al grupo completo de controles.

Así, los gestores de contenido deberían incluir una herramienta que permita identificar grupos de controles comunes por su función o significado, y asignarles un título.

En relación al posicionamiento y ordenación de los campos de formulario, hay que asegurar que el orden del foco u orden de tabulación con teclado por los campos de un formulario, y en general por todos los elementos de interacción, es correcto y tiene sentido. Para ello se han de situar los elementos de interacción y campos de formulario en la ubicación adecuada de forma que el orden de tabulación por defecto ya siga un orden lógico, que tenga sentido.

[6] WCAG 2.1 Input Purposes: https://www.w3.org/TR/WCAG21/#input-purposes
[7] HTML 5.2 Autocomplete: https://www.w3.org/TR/html52/sec-forms.html#autofilling-form-controls-the-autocomplete-attribute

Excepcionalmente, si por alguna circunstancia es necesario cambiar el orden del foco por defecto habrá que hacerlo de forma que éste siga teniendo sentido. Por ejemplo, en algunos casos se desea que el primer elemento en recibir el foco sea el buscador, pero este no se encuentra al comienzo de la página y no es el primer elemento en el orden de tabulación por defecto. Para especificar un nuevo orden lógico de tabulación, en (X)HTML se podrá usar el atributo *tabindex* en aquellos elementos de interacción para los que se quiera modificar su orden, asegurándose siempre que el orden sigue teniendo sentido.

4.13. INCLUSIÓN DE OBJETOS PROGRAMADOS

Un objeto incrustado es un elemento externo que funciona insertado en una página Web a través del elemento OBJECT y que proporciona nuevas características a la misma (efectos decorativos, información, funcionalidades relevantes…).

Cuando se incrusta un objeto se deben tener en cuenta principalmente dos aspectos:

- **La alternativa al objeto**. Todo objeto debe contar con una alternativa que muestre contenido o funcionalidad equivalente, para aquellas situaciones en las que no se disponga de soporte para objetos o éstos se encuentren deshabilitados. La alternativa se introduce en el interior del elemento OBJECT, esto es, entre sus etiquetas de apertura y de cierre, y puede ser un texto o cualquier código (X)HTML+CSS.
- **La accesibilidad del objeto**: Cualquier objeto programado debe ser directamente accesible, ya que puede que el usuario tenga el *plugin* y los objetos activados, por lo que en este caso no accederá a la alternativa, sino al propio objeto, y por lo tanto debe poder manejarlo. Para que un objeto sea directamente accesible, se deben tener en cuenta principalmente los siguientes aspectos:
    - Permitir una navegación coherente dentro del objeto, y entre el objeto y la página que lo contiene.
    - Permitir su uso con independencia del dispositivo de entrada.
    - Permitir una tabulación adecuada.
    - Contener un etiquetado adecuado de los controles del objeto, botones, enlaces, campos de formulario, etc.
    - No incluir destellos o movimientos que no puedan ser controlados.
    - No provocar actualizaciones automáticas periódicas.
    - Poseer niveles de contraste adecuados.

Cuando se introduce un objeto programado a través de un gestor de contenidos, éste debe hacerlo utilizando el elemento OBJECT en lugar del elemento

propietario `EMBED`, el cual no se encuentra recogido en las especificaciones oficiales del W3C.

Ejemplo de código incorrecto 3

```
<object classid="clsid:d27cdb6e-ae6d-11cf-96b8-444553540000"
codebase="http://download.macromedia.com/pub/shockwave/cabs/
flash/swflash.cab#version=6,0,40,0">
<param name="quality" value="high" />
<param name="movie" value="../prueba.swf" />
<embed pluginspage="http://www.macromedia.com/go/getflashplayer"
quality="high"      src="../prueba.swf"      type="application/x-
shockwave-flash"></embed>
</object>
```

Otra opción que debería incluir un gestor de contenidos entre las propiedades del objeto programado es la de poder definir una alternativa al mismo (al menos textual).

Por otro lado, se ha de tener en cuenta que cuando la información se presenta con movimiento se puede dificultar la navegación de personas con discapacidades neurológicas y cognitivas. En este aspecto, sería conveniente que los gestores contaran con funcionalidades que permitan al usuario controlar el movimiento de los objetos programados, como por ejemplo, la posibilidad de anular tanto la ejecución automática del objeto al cargar la página, como su repetición.

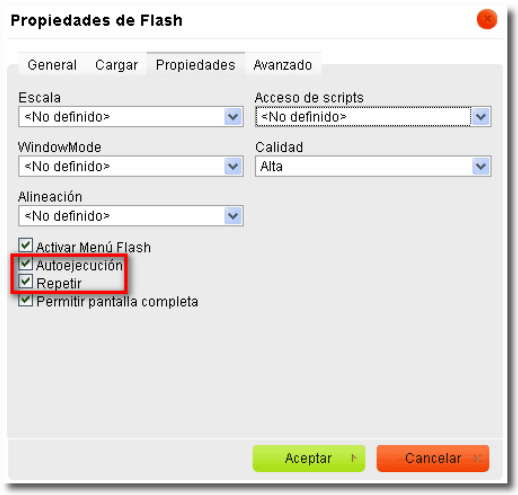

Figura 16. Funcionalidades para el control de movimiento en objetos

4.14. VALIDACIÓN GRAMATICAL

En las WCAG 2.1 ya no es obligatorio que el código valide gramaticalmente. Aunque la validación gramatical no sea un requisito imprescindible, sigue siendo recomendable para asegurar la compatibilidad del código entre diferentes navegadores.

Como requisito mínimo, el código de las páginas debe ser *procesable*. Es decir, no debe haber errores en el código que puedan causar problemas de interpretación a los diferentes navegadores y aplicaciones de usuario.

Para que el código sea procesable se ha de cumplir que al menos esté *bien formado*. La apertura y cierre de las etiquetas debe seguir la especificación. Deben existir etiquetas de cierre para todos los elementos que las requieran y no deben existir para aquellos elementos en los que estén prohibidas. Las etiquetas de apertura y de cierre deben estar anidadas correctamente para todos los elementos. El valor de los atributos debe estar correctamente entrecomillado y no se deben repetir valores en aquellos atributos que requieran un valor único (por ejemplo, los `id`).

Idealmente, y aunque no sea obligatorio, se recomienda que el código de las plantillas (X)HTML utilizadas en los gestores de contenido no contenga errores de validación gramatical. Se ha de tener en cuenta que las reglas varían en función de la gramática (*HTML* o *XHTML*) y versión (*Transitional* o *Strict*) utilizadas. Así, un mismo código puede ser válido para una gramática pero no para otra.

Del mismo modo, el código de las hojas de estilo CSS asociadas a dichas plantillas también se recomienda, si es posible, que sea gramaticalmente válido. En caso contrario, por ejemplo si se emplean propiedades propietarias o experimentales, al menos debería ser sintácticamente correcto asegurando que la presentación entre diferentes navegadores es consistente aunque no necesariamente idéntica.

Para facilitar la tarea de análisis del código (X)HTML y CSS de una página Web, el W3C proporciona herramientas online:

- **Markup Validation Service**: http://validator.w3.org/
- **CSS Validation Service**: http://jigsaw.w3.org/css-validator/

No obstante, una opción interesante sería la de incluir en el propio gestor de contenidos un validador interno, que permita analizar el código (X)HTML y CSS cada vez que se editen los contenidos de una página Web (inserción, modificación o eliminación de textos, tablas, imágenes, objetos, formularios, etc.) indicando aquellos errores que impidan un procesamiento correcto.

4.15. ACCESIBILIDAD DE LA INTERFAZ DE USUARIO

Además de generar contenido accesible, sería recomendable que el gestor de contenidos fuera accesible para las personas con discapacidad de forma que lo puedan usar sin encontrar barreras de acceso. Para ello, el interfaz de usuario del gestor de contenidos debe cumplir los requisitos de las Pautas de Accesibilidad para al Contenido Web (WCAG 2.1) para un nivel de conformidad AA.

A continuación, y a modo de resumen, se describen brevemente algunos de los requisitos de accesibilidad que deben cumplir los gestores de contenidos desarrollados con tecnologías web (HTML, CSS, JavaScript, etc.):

- Todo contenido no textual del interfaz debe tener un equivalente textual accesible para los lectores de pantalla. Así, en el código HTML los iconos de los botones de las barras de herramientas deben tener su texto alternativo o, si se incluyen como imágenes de fondo, disponer de un texto equivalente en el contenido de la página.
- Los controles de formulario usados para generar los elementos de interacción del editor deben tener etiquetas asociadas explícitamente o disponer de un título que identifique su función.
- En los botones, iconos y textos del interfaz de usuario no se debe proporcionar información únicamente mediante el color y las combinaciones de color de primer plano y de fondo deben tener el contraste suficiente.
- Todos los elementos de interacción deben ser operables mediante teclado, bien mediante el tabulador o mediante alguna combinación de teclas específicas. El orden de tabulación debe ser el adecuado y ha de haber un indicador visual de dónde se encuentra el foco del teclado. Si fuera necesario, para facilitar el uso de la herramienta de forma independiente del dispositivo, se proporcionará información de ayuda sobre su manejo mediante el teclado y otros productos de apoyo (atajos de teclado, etc.).
- En caso de emplear JavaScript para crear elementos de interacción avanzados, se han de aplicar las pautas y recomendaciones de la especificación de WAI-ARIA[8] (Accesibilidad para las aplicaciones de Internet enriquecidas), usando las técnicas necesarias para indicar el nombre, rol y valor de los elementos de interacción que no se basan en componentes estándar.

[8] WAI-ARIA: http://www.w3.org/TR/wai-aria/

4.16. ASISTENCIA PARA LA CREACIÓN DE CONTENIDO ACCESIBLE

Finalmente, además de que el gestor de contenido permita crear contenido accesible y que su interfaz sea también accesible de forma que pueda ser usado por personas con discapacidad, es altamente recomendable que estas herramientas den apoyo a los autores para crear contenido accesible.

Por ejemplo, se puede integrar en el gestor de contenidos alguna herramienta de validación de accesibilidad. Mientras se edita el código puede resaltar los problemas que se van detectando, de forma similar a como actúa un corrector ortográfico, o bien puede ser una validación que se invoque a través de un botón u opción de menú. Independientemente de su forma, dicha herramienta comprobará el código del contenido editado detectando los problemas de accesibilidad si los hubiese y proporcionando a los usuarios una descripción de dichos problemas y una ayuda para su resolución.

Por tanto, al igual que otras herramientas de edición como Acrobat Pro u Office 2010 incorporan validadores de accesibilidad para la revisión de los documentos, es recomendable la integración de soluciones que permitan la validación de accesibilidad del contenido web en el momento de su edición y que proporcionen ayuda a los usuarios para la corrección de los problemas detectados.

De esta forma se aumenta la posibilidad de que el contenido contribuido mediante los gestores de contenido sea accesible en el momento de la publicación, disminuyendo así la necesidad o importancia de realizar revisiones periódicas para detectar problemas de accesibilidad en el contenido ya publicado.

5. PERSPECTIVAS ACTUALES Y FUTURAS

Dada su flexibilidad y facilidad de uso, los gestores de contenido se están erigiendo en una de las herramientas preferidas por los editores finales para la creación y mantenimiento de sitios Web dinámicos.

Pese a ello y como se ha podido comprobar a lo largo de la presente guía, el hecho de utilizar gestores de contenido accesibles no implica que los contenidos generados a través de los mismos sean finalmente accesibles. Dado el **carácter permisivo** de los gestores de contenido actuales, para poder garantizar la accesibilidad de un sitio Web es necesario que el **editor final de contenidos** lleve a cabo distintas **revisiones manuales**, que no puede realizar de forma automática el propio gestor.

Hasta que los editores visuales o los gestores de contenido generen código accesible, será necesario un trabajo del **equipo de desarrollo de un sitio web** para modificar el editor y adaptarlo a la creación de contenido accesible.

De cara al futuro, se contempla como posible mejora la incorporación de funcionalidades que permitan crear contenidos atendiendo a su **significado** y no al formato o apariencia final, aumentando de esta forma la **carga semántica**.

Por otro lado, la inclusión de **revisiones automáticas** permitiría que no recayera todo el control de la accesibilidad del sitio Web sobre el editor final, que puede tener escasos o nulos conocimientos en la materia.