@dministración electrónica

Observatorio de Accesibilidad

GUÍA DE ACCESIBILIDAD EN SEDES ELECTRÓNICAS

(Versión 3, junio 2023)

GOBIERNO DE ESPAÑA
MINISTERIO DE ASUNTOS ECONÓMICOS Y TRANSFORMACIÓN DIGITAL
MINISTERIO DE HACIENDA Y FUNCIÓN PÚBLICA

Guía de accesibilidad en sedes electrónicas

(Versión 3, junio 2023)

Elaboración y coordinación de contenidos:
Secretaría General de Administración Digital (SGAD)

Publicación disponible en el Portal de Administración Electrónica (PAe)
http://administracionelectronica.gob.es

Edita:

Secretaría General Técnica
Centro de Publicaciones

Subdirección de Información, Documentación y Publicaciones
Secretaría General Técnica
Centro de Publicaciones

Colección: *Administración electrónica*

NIPO (MINECO): 094-23-059-6

NIPO (MHFP): 137-23-084-4

ÍNDICE

Tabla 1 Control de cambios

Versión 3	Descripción de cambios
Enero 2023	Actualización de UNE-EN 301549:2019 por UNE-EN 301549:2022 Actualización de EN 301549 v2.1.2 (2018-08) por EN 301549 v3.2.1 (2021-03)
Junio 2023	Logo Gobierno de España. Vicepresidencia Primera del Gobierno. Ministerio de Asuntos Económicos y Transformación Digital. Actualización del formato de estilos del documento. Modificación de los textos alternativos de las imágenes y actualización de algunos contenidos.

ÍNDICE DE IMÁGENES

1. INTRODUCCIÓN

En la **Ley 40/2015, de 1 de octubre, de Régimen Jurídico del Sector Público** se define el concepto de "sede electrónica" como aquella dirección electrónica, disponible para los ciudadanos a través de redes de telecomunicaciones, cuya titularidad corresponde a una Administración Pública y que conlleva la responsabilidad del titular respecto de la integridad, veracidad y actualización de la información y los servicios a los que pueda accederse a través de la misma.

La implementación de cualquier información, servicio o mecanismo de tramitación incluido en una Sede electrónica sin tener en cuenta los requisitos de accesibilidad puede ocasionar una importante barrera de acceso a determinados usuarios, por lo que se hace indispensable tener presentes los aspectos relacionados con la **accesibilidad** si se desea desarrollar una Sede electrónica utilizable por cualquier ciudadano.

2. OBJETIVO DE LA GUÍA

El objetivo de la presente guía es el de ayudar a los técnicos responsables del desarrollo de Sedes electrónicas en la tarea de incluir **mecanismos de tramitación electrónica**, de tal forma que éstos sean **accesibles** según los requisitos que dictan las normativas vigentes, permitiendo así el acceso a ellos a todos los usuarios, independientemente de sus propias limitaciones o las derivadas de su entorno.

La normativa de accesibilidad en la que se basa esta nueva versión de la guía es la recomendación **WCAG 2.1** de junio de 2018, en concreto esta guía recoge los criterios de conformidad de nivel A y AA de WCAG 2.1, a los que hace referencia la norma **EN 301 549 v3.2.1:2022**, declarada **estándar armonizado** por la comisión europea en diciembre de 2018, y por tanto es el estándar que aplica a las administraciones públicas españolas para el cumplimiento del **Real Decreto 1112/2018**.

La guía se focaliza en los aspectos específicos de accesibilidad en las sedes electrónicas. Por lo tanto, se dedican los mayores esfuerzos a la identidad y firma electrónica y a los formularios electrónicos. También se resumirán de forma rápida algunas cuestiones genéricas de accesibilidad que aplican a las sedes electrónicas al igual que a cualquier otro sitio web.

Se adjunta con la guía un ejemplo de **formulario accesible** en formato XHTML+CSS para su posible uso como modelo de referencia.

3. ACCESIBILIDAD EN SISTEMAS DE AUTENTICACIÓN Y CERTIFICACIÓN

En el desarrollo actual de aplicaciones seguras para la tramitación de procedimientos en Sedes electrónicas de la Administración Pública, se hace imprescindible asegurar la **autenticidad** del usuario, la **integridad** de los datos intercambiados, y el **no repudio** de las transacciones realizadas.

En este sentido, surgen soluciones como la **firma digital** y los certificados digitales (**DNI electrónico, FNMT, etc.)** que se basan en la utilización de sistemas criptográficos para alcanzar los objetivos de seguridad. Los citados sistemas criptográficos se materializan en certificados (claves) a los cuales es necesario acceder desde el ordenador personal del usuario que desea realizar la operación. Bien, porque la tarjeta criptográfica se haya introducido en el lector de tarjetas del ordenador o bien porque el certificado software esté almacenado en el navegador.

Figura 1. DNI Electrónico

Para **la operación con estos certificados** dentro de una Sede electrónica se debe utilizar **tecnología de ejecución dinámica**, de mayor potencia que el lenguaje (X)HTML estándar. Dicha tecnología consiste en la utilización de scripts y aplicación de una **lógica de cliente** basada en applets, objetos ActiveX u otros elementos que puedan ser ejecutados en el ordenador personal del usuario en el momento de la operación con el certificado.

Anteriormente para cumplir con las WCAG 1.0, este empleo de objetos programados ejecutados en el navegador del usuario chocaba directamente con el requisito de accesibilidad que exigía que las páginas debían poder utilizarse, aunque los scripts y objetos de programación estuviesen desconectados o no fuesen soportados. Se consideraba una barrera tecnológica que no era posible superar, dado que **no hay forma de leer las claves de un usuario prescindiendo de este tipo de tecnologías**. Por tanto, esta limitación tecnológica se trataba como una excepción y no se consideraba un obstáculo para que la Sede electrónica fuese finalmente accesible.

En cambio, bajo los nuevos requisitos de las WCAG 2, esto deja de ser un problema y no tiene por qué tratarse como una excepción. En la nueva normativa se permite el uso de tecnologías y elementos de programación, sin necesidad de que se siga proporcionando la misma funcionalidad cuando no se soporte dicha tecnología, siempre que la misma se emplee de forma compatible con la accesibilidad.

Se entiende que una tecnología (por ejemplo, scripts, applets, etc.) es compatible con la accesibilidad si con la misma es posible crear contenido accesible y los navegadores y productos de apoyo (p. ej. lectores de pantalla) de uso común son compatibles con dicha tecnología o, si es necesario un plugin, éste está disponible para su descarga o compra con la misma facilidad y precio para las personas con discapacidad como para una persona sin discapacidad.

Así, por ejemplo, la tecnología **JavaScript** se considera que es compatible con la accesibilidad y se puede utilizar siempre y cuando se haga de forma no intrusiva y accesible. Por ejemplo, si se genera contenido dinámicamente éste debe ser accesible, se debe preservar siempre el orden de lectura del contenido y todos los elementos de interacción deben ser accesibles con teclado y en el orden de tabulación correcto, entre otras cosas.

En caso de que se emplee **JavaScript** para crear funcionalidades e interfaces de usuarios complejas se deben emplear las pautas indicadas en la especificación de WAI-ARIA del W3C para añadir la capa de accesibilidad necesaria para asegurar su compatibilidad con los productos de apoyo. Para ampliar información al respecto se recomienda consultar la especificación de WAI-ARIA así como también el documento introductorio WAI-ARIA Primer.

Sin embargo, siempre es recomendable que frente a la posibilidad de realizar la citada operación online se permitan otras **alternativas** como por ejemplo posibilitar la realización presencial de la operación, incluir un formulario PDF accesible firmado digitalmente, posibilitar la tramitación telefónica, por correo postal, etc.

Además, en la **declaración de accesibilidad** de la Sede electrónica se han de identificar las tecnologías de las que se depende para poder acceder a los contenidos de forma accesible y poder garantizar el completo acceso de los usuarios a los procesos de tramitación y gestión electrónica, así como otros requisitos que fuesen necesarios para su uso.

3.1. APPLETS DE FIRMA

En el caso concreto de los **applets de firma**, estos deben ser incluidos únicamente por medio del elemento `OBJECT`, ya que el elemento `APPLET` está

desaconsejado, y su **interfaz** (si disponen de ella) debe ser directamente accesible, para lo cual se deben cumplir los siguientes condicionantes:

- Permitir una navegación coherente dentro del applet, y entre el applet y la página que lo contiene.
- Permitir su uso con independencia del dispositivo de entrada (ratón, teclado, etc.)
- Contener un etiquetado adecuado de los controles del applet, botones, enlaces, campos de formulario, etc.
- Permitir una tabulación entre controles adecuada.
- No incluir destellos o movimientos que no puedan ser controlados.
- No provocar actualizaciones automáticas periódicas.
- Poseer niveles de contraste adecuados.

En relación a los applets desarrollados en Java, cabe destacar la existencia de Java Access Bridge, un paquete de gran utilidad que permite a las herramientas y tecnologías de apoyo acceder a las propiedades de accesibilidad que pueden ser ofrecidas en los diversos elementos gráficos que forman la interfaz de una aplicación Java (desarrollada con una versión 1.4.1 o superior del JDK de Java). Además, es aconsejable que la interfaz de este tipo de applets sean desarrollada mediante componentes de Java que implementen el Package javax.accessibility (como Swing), ya que de esta forma los productos de apoyo serán capaces de obtener esa información a través de Java Access Bridge permitiendo que un lector de pantalla obtenga la información y la lea en voz alta, o un magnificador de pantalla sepa cómo manejar los componentes del Applet para aumentarlo, etc.

4. ACCESIBILIDAD EN FORMULARIOS

Los formularios constituyen los principales elementos de interacción entre los usuarios y las Sedes electrónicas a la hora de utilizar sus servicios telemáticos, por lo que es muy importante garantizar la accesibilidad de estos.

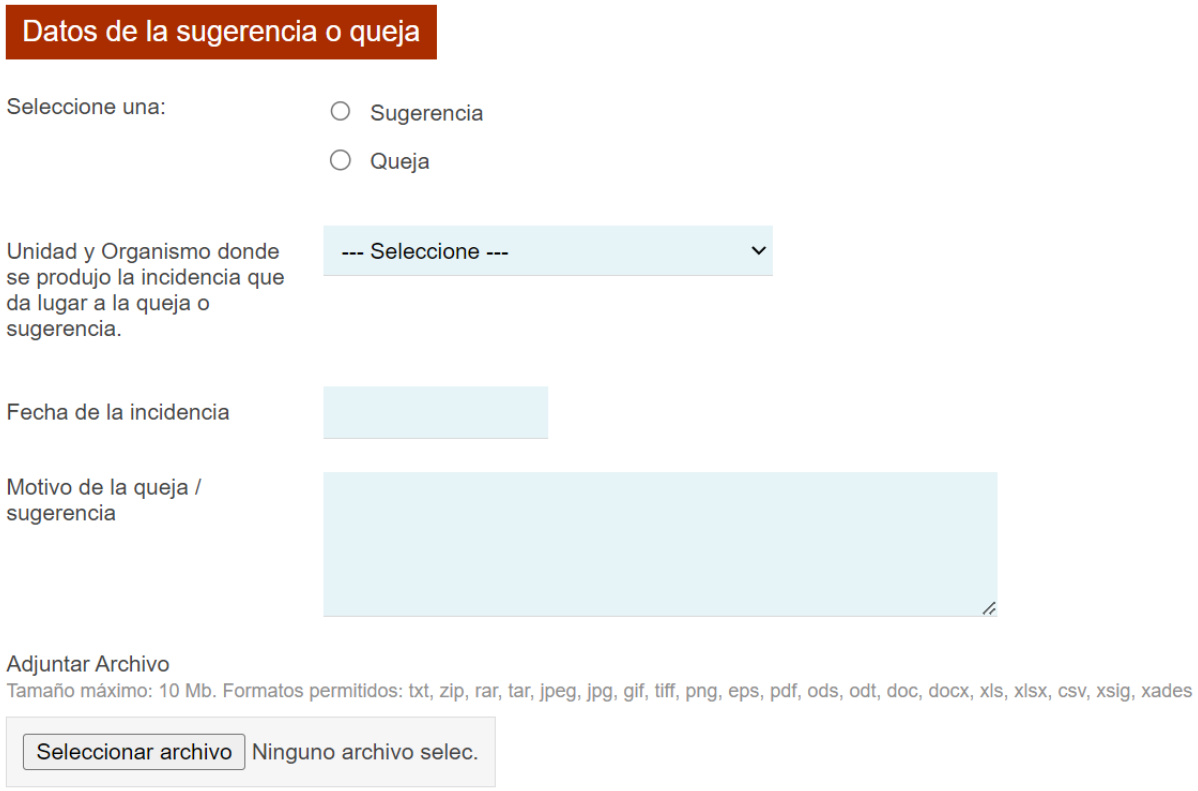

Figura 2. Formulario Gestión de Sugerencias o quejas de la Sede Electrónica

Para garantizar la accesibilidad de los formularios, se deberán cumplir los siguientes condicionantes:

- Asociación de etiquetas con sus respectivos controles.
- Inclusión de la etiqueta en el nombre accesible del campo de formulario.
- Identificación del propósito de las entradas.
- Agrupación de controles de formularios.
- Identificación, si es posible, de los pasos de un proceso (no imprescindible, aunque sí recomendable).

- Identificación de errores.
- Ayuda e instrucciones.
- Sugerencias ante errores.
- Prevención de errores.
- Independencia de dispositivo.
- Predictibilidad ante cambios de contexto.

4.1. ELEMENTO FORM

Todo formulario debe estar incluido dentro de un elemento de HTML `FORM`.

En caso de que una página web tenga varios formularios con propósitos distintos, por ejemplo, un formulario de búsqueda y un formulario de contacto, se recomienda utilizar elementos `FORM` independientes, uno para cada formulario, de forma que el código y el procesamiento del formulario sean más sencillos.

Ejemplo de código 1

```
<form action="form1.php" method="post">
<div>
   <p><label for="busqueda">Texto a buscar: <input type="text"
            name="busqueda" id="busqueda" /></label></p>
   <p><input type="submit" value="Buscar" name="submit" /></p>
</div>
</form>
<form action="form2.php" method="post">
<div>
   <p><label for="nombre">Nombre <input type="text"
            name="nombre" id="nombre" /></label></p>
    <p><label for="email">E-mail <input type="text"
            name="email" id="email" /></label></p>
   <p><label for="comentarios">Comentarios <textarea na
            me="comentarios" cols="40" rows="5"
            id="comentarios"
            class="textarea"></textarea></label></p>
   <p><input type="submit" value="Enviar" name="submit" /></p>
</div>
</form>
```

4.2. BOTÓN DE ENVÍO

Para que un formulario pueda ser enviado correctamente, además de estar definido mediante el elemento `FORM` y poseer una URL válida en el atributo

action, ha de contar con un botón de envío generado con un elemento INPUT de tipo *submit* o *image*.

Se recomienda la utilización de elementos INPUT tipo *submit*. En caso de emplear un botón de envío de tipo *image* (imagen en lugar de texto), es necesario proporcionar un texto alternativo equivalente que permita comprender la funcionalidad del botón a los usuarios que navegan sin imágenes o a los usuarios de lector de pantalla.

Ejemplo de código 2

```
<input type="image" src="../imgs/enviar.png" alt="Enviar" value="Enviar" />
```

4.3. ASOCIACIÓN DE ETIQUETAS CON SUS RESPECTIVOS CONTROLES

Para realizar una asociación correcta de las etiquetas de formulario con sus respectivos controles se deben asociar explícitamente. Además, todas las etiquetas de un formulario deben estar marcadas mediante el elemento LABEL.

Como excepción, cuando no existe un texto visible en la página que sirva como etiqueta de un control y que se pueda marcar como LABEL, entonces se admite usar el título del campo de formulario a modo de etiqueta para indicar su función.

4.3.1. Asociación explícita

La **asociación explícita** permite que los agentes de usuario relacionen automáticamente las etiquetas con sus controles de formulario, para ello se debe utilizar el atributo *id* en los controles del formulario y el atributo *for* en las etiquetas. El contenido de ambos atributos debe ser igual para cada par etiqueta-control.

Ejemplo de código 3

```
<label for="nombre">Nombre:</label>
<input type="text" name="nombre" id="nombre"/>
```

Siempre que sea posible, utilice el elemento LABEL para asociar texto con elementos de formulario de forma explícita.

4.3.2. Asociación implícita

En algunas situaciones, los controles de formulario no se pueden etiquetar explícitamente. Por ejemplo, es posible que un autor de contenido no conozca el *id* del campo de un formulario generado por una secuencia de comandos, o que esa secuencia de comandos no agregue ningún *id*. En este caso, el elemento LABEL se usa como contenedor tanto para el control de formulario como para el texto de la etiqueta, de modo que los dos se asocian implícitamente. En general, las etiquetas explícitas están mejor respaldadas por tecnología de asistencia.

Ejemplo de código 4

```
<label>Nombre:
<input type=”text” name=”nombre”/></label>
```

Si no hay ninguna etiqueta LABEL, o si el control de formulario no está asociado implícita o explícitamente con alguna etiqueta, un lector de pantalla leerá algo así como «Editar texto en blanco», lo cual no es de mucha ayuda.

4.3.3. Campos obligatorios en etiquetas

Cuando en el formulario hay campos obligatorios se debe informar al principio del formulario y en cada campo de formulario requerido se debe indicar dentro del elemento LABEL.

Para ello, utiliza el elemento *abbr* para abreviar la palabra “asterisco” y mediante la propiedad *aria-label* de WAI-ARIA informamos a los usuarios de tecnologías de apoyo del significado del “*”.

Ejemplo de código 5

```
<p> Los campos marcados con asterisco <abbr aria-label="Obligato-
rio">*</abbr> son obligatorios</p>

<label for=”nombre”><abbr aria-label="Obligatorio">*</abbr> Nom-
bre: </label>
<input type=”text” name=”nombre” id=”nombre”/>
```

4.3.4. Ocultar el texto de la etiqueta

Una etiqueta en un control de formulario ayuda a todos a comprender mejor su propósito. En algunos casos, el propósito puede ser lo suficientemente claro a partir del contexto cuando el contenido se representa visualmente. La etiqueta LABEL se puede ocultar visualmente, aunque aún debe proporcionarse dentro del código para admitir otras formas de presentación e interacción, como lectores de pantalla y usuarios de entrada de voz.

En el siguiente ejemplo, el campo de búsqueda se coloca directamente al lado del botón de búsqueda. El propósito del campo de entrada de texto es evidente por el contexto en la mayoría de las situaciones.

Figura 3. Botón del buscador

Ejemplo de código 6

```
<label  for="buscar" class="visuallyhidden"> Buscar: </label>
<input name="buscar" id="buscar" type="text">
<button type="submit">Buscar</button>

<!-- código CSS para ocultar el elemento <label> a los usuario vi-
dentes y accesible para las tecnologías de apoyo -->

/* Clase para ocultar elementos en CSS */

.visuallyhidden {
  border: 0;
  clip: rect(0 0 0 0);
  height: 1px;
  margin: -1px;
  overflow: hidden;
  padding: 0;
  position: absolute;
  width: 1px;
}
```

En este enfoque, el elemento `LABEL` se proporciona para identificar un control de formulario dentro del código, pero se oculta visualmente para evitar la redundancia para los usuarios que pueden entender el propósito de las señales visuales. Otra opción es hacer uso de las propiedades *aria-label y aria-labelledby* de WAI-ARIA.

Ejemplo de código 7

```
<input name="buscar" aria-label="Buscar" type="text">
<button type="submit">Buscar</button>
```

Ejemplo de código 8

```
<input name="buscar" aria-labelledby="button" type="text">
<button type="submit" id="button">Buscar</button>
```

4.3.5. Etiquetas visibles en grupo de controles

Cuando un campo de entrada de datos necesita varios controles de formulario la agrupación debe realizarse visualmente y en el código, por ejemplo, utilizando una etiqueta `LABEL` visible que informe del campo a cumplimentar.

Además, como se verá en el apartado de "Agrupación de formularios", se utilizarán los elementos `FIELDSET` y `LEGEND` para asociar los controles de formulario relacionados.

En el siguiente ejemplo se usan tres campos para solicitar una fecha (día, mes y año) y utilizan el atributo *`title`* para identificar cada control en vez de incluir una etiqueta `LABEL` para cada uno.

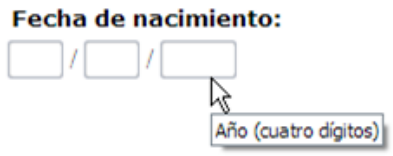

Figura 4. Atributo "title" en vez de etiquetas "label"

Este enfoque generalmente es menos confiable y **no se recomienda** porque algunos lectores de pantalla y tecnologías de apoyo no interpretan el atributo *`title`* como un reemplazo de la etiqueta del elemento, posiblemente porque el atributo *`title`* se usa a menudo para proporcionar información no esencial.

La información del atributo *`title`* se muestra a los usuarios visuales cuando se pasa el ratón sobre el campo del formulario pero no es leída por los usuarios de teclado cuando el campo de entrada recibe el foco del teclado. Incumpliendo el requisito **9.3.3.2 Etiquetas e Instrucciones** en las que las etiquetas siempre deben estar visibles, ya que brinda asistencia a todos los usuarios que necesitan ayuda para comprender el propósito del campo.

En este otro ejemplo, se informa al usuario del formato permitido para el campo "Día" y "Año" a través del atributo *`placeholder`* que **no se recomienda** porque algunos lectores de pantalla y tecnologías de apoyo no interpretan este atributo.

Figura 5. Atributo "placeholder" en vez de etiquetas "label"

Nuestra recomendación es utilizar elementos `LABEL` visibles junto a sus controles asociados en los que se informe a todos los usuarios de los datos y formatos a introducir para que los usuarios eviten errores y no tengan dificultades al introducir los datos solicitados.

Figura 6. Ejemplo Agrupación correcta de controles

4.4. INCLUSIÓN DE LA ETIQUETA EN EL NOMBRE ACCESIBLE DEL CAMPO DE FORMULARIO

En relación con el apartado anterior y el etiquetado, en los componentes del interfaz de usuario (como los campos de formulario o botones) es necesario que el **texto visible que actúa como su etiqueta** y que sirve para reconocerlos visualmente también **forme parte de su nombre accesible**.

Entendemos por **nombre accesible** el que emplean las diferentes herramientas software para identificar e informar a los usuarios acerca de los componentes, así como para interactuar con los mismos. Por ejemplo, es el nombre que va a leer un lector de pantalla cuando informe acerca del componente y es el nombre que empleará un sistema de reconocimiento de voz para activar el componente.

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 interactúa mediante **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 entonces la interacción resulta mucho más complicada para este tipo de personas y se abre la posibilidad además de accionar componentes accidentalmente.

El **nombre accesible debe contener sin modificación el texto visible**. Se considera como error el hecho de que el nombre accesible no contenga el texto visible de la etiqueta o que el nombre accesible contenga al texto visible pero en un orden diferente o con otras palabras intercaladas. Se recomienda no utilizar los atributos de WAI-ARIA relacionados con el nombre accesible cuando el componente de la interfaz tiene un texto visible que actúa como su etiqueta, ya que sería redundante e innecesario.

Si aplicamos todas las indicaciones de las pautas a la hora de proporcionar un etiquetado correcto para los campos de formulario u otros elementos de interacción ya estamos cumpliendo este requisito. Es decir, el atributo o mecanismo nativo de HTML para asignar un nombre a un componente es el nombre accesible que emplearán los lectores de pantalla o sistemas de reconocimiento de voz.

Por ejemplo, si en los formularios estamos empleando la asociación explícita entonces 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`.

Ejemplo de código 9

```
<label for="nombre">Nombre: <input type="text" id="nombre" > </label>
```

En el caso de los `INPUT` de tipo *`submit`* el nombre accesible por defecto es el valor de su atributo *`value`*.

Ejemplo de código 10

```
<input type="submit" value="Buscar">
```

En el caso de los botones el nombre accesible es su contenido textual.

Ejemplo de código 11

```
<button type="submit">Buscar</button>
```

En el caso de que estemos en otras situaciones, como cuando tenemos controles personalizados realizados mediante scripts, entonces podemos hacer uso de las propiedades de WAI-ARIA como *`aria-label`*, *`aria-labelledby`* o *`aria-describedby`* para asignar un nombre accesible a los componentes cuando no llevan el texto de la etiqueta visible.

Ejemplo de código 12

```
<div role="button" aria-label="Buscar Tramitaciones" class="button image" tabindex="0" aria-pressed="false">Buscar</div>
```

Nota: En el ejemplo anterior se ha utilizado el patrón de diseño estándar de WAI-ARIA para componentes personalizados.

Si utilizamos *`aria-label`* o *`aria-labelledby`* en controles de formulario, que ya tienen una etiqueta `LABEL` visible, el **nombre accesible** del campo será sustituido por el que tengan las propiedades *`aria-label`* o *`aria-labelledby`*., ocurriría lo mismo en botones y enlaces.

Ejemplo de código desaconsejado 1

```
<label for="nombre" aria-label="Campo obligatorio">Nombre: <input type="text" id="nombre"></label>
```

En este caso el texto "Nombre" de la etiqueta visible no coincide con el texto "Campo obligatorio" de la propiedad *aria-label* de modo que el usuario de lector de pantalla no sabrá que ha de introducir el nombre. Del mismo modo el usuario que interactúa mediante entrada de voz cuando diga "Nombre" no podrá acceder al campo de entrada. Fallaría el requisito **9.2.5.3 Inclusión de la etiqueta en el nombre**.

Por lo tanto, se desaconseja utilizar la propiedad *aria-label* en controles de formulario, botones y enlaces que ya disponen de un texto visible

4.5. IDENTIFICACIÓN DEL PROPÓSITO DE LAS ENTRADAS

En las UNE-EN 301459:2022 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 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, sirven para indicar que un campo se trata de un email o un teléfono, pero no aclaran cuál es el dato concreto al que se refieren (¿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 junto con un valor que indique el tipo de información solicitada. Por ejemplo, algunos de los valores posibles para el atributo *autocompleten* 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.).

En el siguiente ejemplo se puede ver un ejemplo del uso del atributo *autocomplete* para solicitar el nombre, los apellidos y la dirección en un formulario.

Ejemplo de código 13

```
<form method="post" action="paso2">
  <div>
    <label for="nombre">Nombre</label>
    <input id="nombre" type="text" autocomplete="given-name" ... >
  </div>
  <div>
    <label for="apellidos">Apellidos:</label>
    <input id="apellidos" type="text" autocomplete="family-name"
      ... >
  </div>
  <div>
    <label for="direccion">Dirección:</label>
    <input type="text" id="direccion" autocomplete="street-ad-
dress"
      ... >
  </div>
  <div>
    <input type="submit" value="Continuar..." >
  </div>
</form>
```

4.6. AGRUPACIÓN DE FORMULARIOS

La agrupación de controles de formulario relacionados hace que los formularios sean más comprensibles para todos los usuarios, ya que los controles relacionados son más fáciles de identificar. También facilita que las personas se concentren en grupos más pequeños y más manejables en lugar de tratar de comprender todo el formulario a la vez.

La agrupación debe realizarse visualmente y en el código, por ejemplo, utilizando los elementos `FIELDSET` y `LEGEND` para asociar los controles de formulario relacionados. El elemento `FIELDSET` proporciona un contenedor para los controles de formulario relacionados y `LEGEND` actúa como un encabezado para identificar el grupo, como se aprecia en la siguiente imagen:

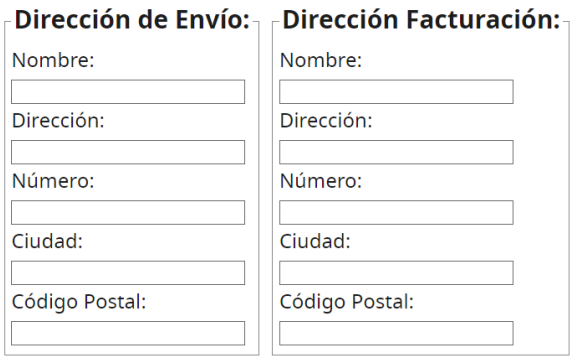

Figura 7. Agrupación de controles de formulario

Se debe **agrupar la información** cuando sea natural y apropiado, por lo tanto, cuando existan controles de formulario relacionados que se puedan agrupar en unidades lógicas, se debe utilizar el elemento `FIELDSET` y aplicar una etiqueta a esas unidades con el elemento `LEGEND`.

Ejemplo de código 14

```
<form action="http://prueba.com/nuevousuario" method="post">
   <fieldset>
   <legend>Dirección de Envío</legend>
   <label for="nombre">Nombre: </label>
   <input type="text" id="nombre" name="nombre" />
   <label for="direccionEnvio">Dirección: </label>
   <input type="text" id="direccionEnvio" name="direccionEnvio" />

   ...más datos personales...

   </fieldset>

   <fieldset>
   <legend>Dirección Facturación</legend>

   ...datos de la dirección de facturación...

   </fieldset>
</form>
```

Cuando se deseen incluir **listas de selección largas** en los menús (en las cuales puede resultar difícil orientarse), se deberán agrupar los elementos de la lista (definidos mediante el elemento `OPTION`) en grupos con el elemento `OPTGROUP`. Se deberá especificar una etiqueta para el grupo de opciones mediante el atributo *`label`* en el elemento `OPTGROUP`.

Ejemplo de código 15

```
  <label for"requisitos">Seleccione el requisito deseado: </label>
  <select id="requisitos" name="PrioridadesAccesibilidad">
     <optgroup label="Prioridad 1">
       <option value="4.2.1">Legibilidad sin CSS</option>
       <option value="4.3.1">Títulos para los marcos</option>
       <option value="4.4.2">Lenguaje claro y sencillo</option>
     </optgroup>
     <optgroup label="Prioridad 2">
       <option value="4.1.4">Metainformación</option>
       <option value="4.3.7">Listas</option>
       <option value="4.5.9">Orden de tabulación</option>
     </optgroup>
   </select>
```

4.7. IDENTIFICACIÓN DE LOS PASOS DE UN PROCESO

En algunos casos puede ser necesario plantearse dividir una operación en distintos pasos, ya sea por ejemplo porque se trata de operaciones largas y complejas que conviene dividir en distintas fases para facilitar el proceso, o bien porque son operaciones variables en las que los datos de cada paso dependerán de los datos introducidos en los pasos anteriores.

En primer lugar, habría que plantearse reducir el número de campos al mínimo imprescindible, siguiendo estrategias como pedir únicamente la información absolutamente necesaria, infiriendo la información a partir de otra ya disponible cuando sea posible y evitar siempre pedir la misma información dos o más veces. No obstante, en algunas ocasiones seguirá siendo necesario solicitar un número elevado de datos.

En estas ocasiones es posible optar por dividir el formulario en varias etapas, preferiblemente no más de cuatro o cinco, que faciliten la tarea y en las que se vayan solicitando los datos relevantes para cada momento.

Es también recomendable que a la vez se establezca un mecanismo de identificación que indique claramente en que parte del proceso nos encontramos en cada momento, así como mantener la información ya proporcionada, dejando que el usuario avance o retroceda por los pasos libremente y sin perder los datos ya introducidos.

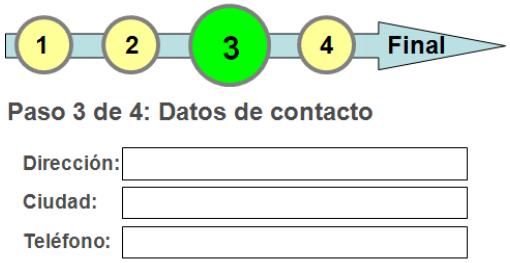

Figura 8. En un formulario de varios pasos indicaremos cuál es el paso actual

4.8. IDENTIFICACIÓN DE ERRORES

Es importante reducir el número de errores importantes o irreversibles que se pueden producir al usar los formularios e informar y ayudar a los usuarios a corregir los errores en caso de que se produzcan.

Si bien nadie está libre de cometer algún error a la hora de interactuar con un sitio web o al introducir datos en un formulario, las personas con discapacidad confían en sus herramientas de apoyo, si un campo de entrada de datos no tiene

asociado correctamente un error de validación un usuario de lector de pantalla tendrá más dificultades para detectarlo y corregirlo.

Se debe informar a los usuarios acerca de todos los **errores de validación** producidos al introducir datos en un formulario. Dichos avisos se deberán mostrar de manera accesible y **en formato de texto** antes del formulario o justo al lado de campo dónde se produce el error, de forma que no pasen inadvertidos para el usuario. Un formulario que sólo indique los campos en los que se han producido errores mediante una marca o un cambio del color del campo no es suficiente. Por ejemplo, un usuario de un lector de pantalla se encontraría con los siguientes problemas:

- No sabría que se ha producido un error hasta que el lector de pantalla no encuentre un texto que le informe dónde se ha producido el error.
- Podría abandonar el formulario, si no detecta dónde está el error, pensando que el formulario no es funcional.

Para indicar los errores se pueden usar imágenes, colores, estilos de texto, etc., pero siempre de forma complementaria a la información proporcionada en formato textual.

A la hora de identificar los errores que se han producido en un formulario se debe:

- Proporcionar descripciones textuales que identifiquen los campos no completados. Para que la información sea clara y la puedan entender todos los usuarios se ha de proporcionar en forma de texto, considerándose insuficiente el incluir únicamente marcas como asteriscos o indicaciones de color.

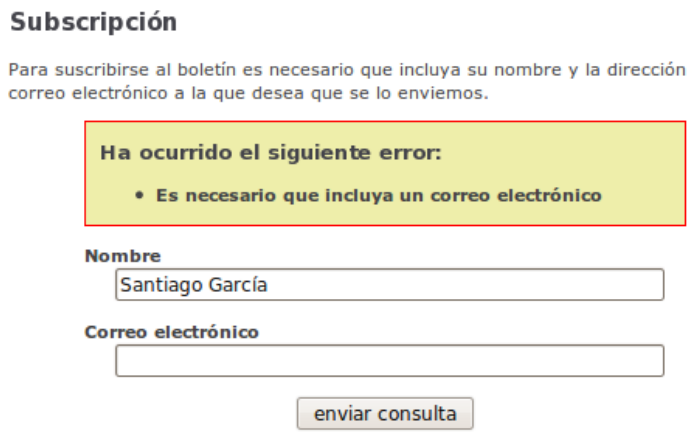

Figura 9. La información sobre los campos que faltan se proporciona en texto

- Proporcionar un texto descriptico que indique al usuario que ha introducido un dato que no cumple el formato requerido o que no se encuentra entre los valores permitidos.

Cuando los mensajes de error se muestran de forma dinámica en un lugar de la página diferente de la ubicación actual del usuario (p. ej. según se está escribiendo en un campo o justo al pulsar el botón de envío sin llegar a enviar los datos del formulario) entonces es importante que estos mensajes se identifiquen como regiones activas de WAI-ARIA tal y como se indica en el apartado 5.7.4 Mensajes de estado. De esta forma los lectores de pantalla podrán reconocer estos mensajes y transmitidos automáticamente al usuario sin que éste tenga que desplazarse desde su posición.

4.9. AYUDA E INSTRUCCIONES

Las instrucciones de un formulario deben indicar cómo se tienen que rellenar los diferentes campos de entrada de datos: campos obligatorios, valores posibles, formato de los datos, etc. para enviar posibles errores. Por tanto, se debe proporcionar la información que sea necesaria para que los usuarios puedan rellenar los formularios correctamente sin cometer errores.

Esta información se ha de proporcionar de forma clara y precisa, en su justa medida. Demasiada información, o información ambigua o imprecisa, puede llegar a ser contraproducente al generar confusión.

En general los mensajes de ayuda se han de usar con precaución limitándolos preferiblemente a algunos casos específicos como:

- Tipos de datos que son de uso menos común, como por ejemplo un código de uso específico para la aplicación actual.
- Información sobre la obligatoriedad de los campos o formatos específicos o preferentes a la hora de introducir los datos.
- Información relativa a la privacidad o seguridad de los datos, como por ejemplo la utilización de los correos electrónicos.

Todo mensaje de ayuda deberá ser redactado con mucha precaución y de la forma más clara y concisa que sea posible. Además, deberán presentarse de forma claramente destacada y que contraste con el resto de información.

Una regla general para cualquier tipo de información de ayuda es que esté claramente asociada a los campos correspondientes, evitando por ejemplo mostrar este tipo de mensajes en un documento aparte. Si se hace de esta manera estaría obligando al usuario a recordar toda la información, perdiendo así eficacia en la retroalimentación al usuario y la interacción con el formulario.

4.9.1. Campos obligatorios

Partiendo de la base de que lo ideal es minimizar el número de campos opcionales que se requieran para evitar añadir ruido innecesario en el proceso que se esté llevando a cabo, existen dos estrategias diferenciadas a la hora de distinguir entre los campos de obligatoria cumplimentación y los opcionales en un formulario.

Ambas estrategias consisten en indicar cuáles son los campos obligatorios o cuáles son los campos opcionales, dejando el último tipo sin ninguna identificación especial. Sin embargo, en general es más conveniente identificar aquellos campos que constituyan la excepción, es decir, si la mayoría de los campos son obligatorios se indicarían los opcionales, y si la mayoría son los opcionales entonces señalaríamos los obligatorios.

También hay que tener en cuenta que se ha de valorar la solución de forma global en el sitio web, y una vez se haya tomado una decisión debe aplicarse a todos los formularios de forma consistente en todo el sitio web.

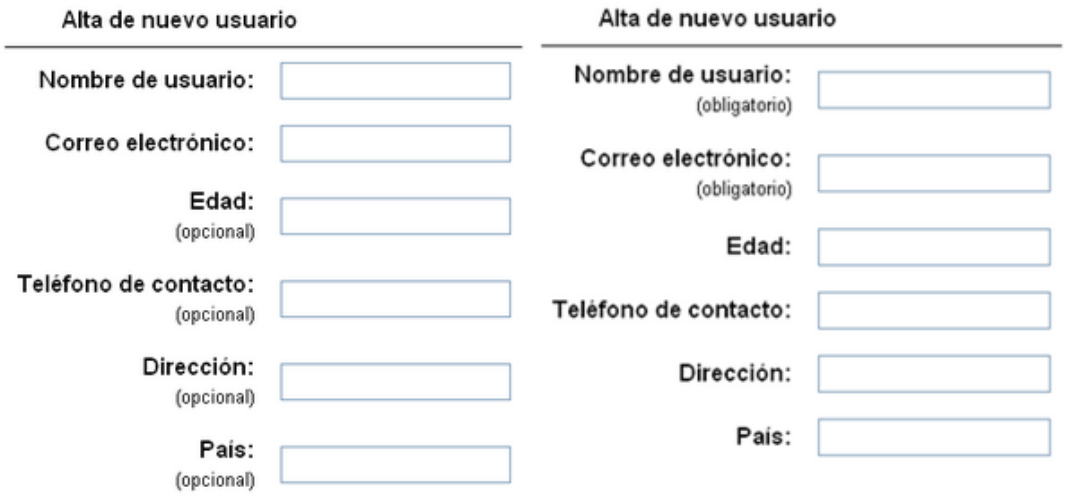

Figura 10. Es preferible indicar aquellos campos que constituyan una excepción a lo general

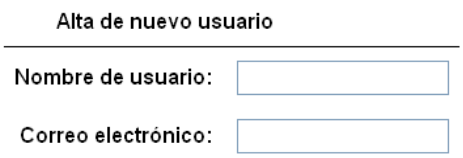

Figura 11. La mejor opción es minimizar el número de campos opcionales

En cuanto a la forma de indicar cuándo un campo es obligatorio u opcional es preferible utilizar una forma textual explícita, por ejemplo *[obligatorio]*, no obstante el uso del símbolo * (asterisco) está bastante extendido y va camino de convertirse en un estándar *de facto*, por lo que es una opción válida siempre que se proporcione una nota explicativa antes del formulario indicando que los campos con asterisco son obligatorios.

Figura 12. Campos obligatorios en un formulario

En este caso, en la etiqueta de cada campo obligatorio se colocará un asterisco. Dicho asterisco se podría marcar como una abreviatura cuyo atributo `aria-label` será "*Obligatorio*" por ejemplo.

Ejemplo de código 16

```
<form action="formulario.html" method="post">
  <p>Nota: Los campos marcados con un asterisco <abbr aria-label="Obligatorio">*</abbr> son obligatorios</p>
  <label for="nombre">
    <abbr aria-label="Obligatorio">*</abbr> Nombre: </label>
    <input type="text" id="nombre" name="nombre" />
  </form>
```

Por último, si en un formulario todos los campos son obligatorios no es necesario indicarlo explícitamente en cada uno de ellos, sino que se podría utilizar un mensaje general al inicio del formulario.

4.9.2. Información de ayuda

Cuando los datos deben respetar un determinado formato se debe informar a los usuarios acerca de las restricciones de dicho formato en las etiquetas de los controles. Es recomendable, además, usar ejemplos para facilitar la comprensión de los formatos.

Por otra parte, en los casos para los que es habitual usar diferentes formatos (por ejemplo, el formato de las fechas y horas puede variar según el país) es recomendable dar varias opciones para que los usuarios escojan el formato de su preferencia.

Si se desea mostrar alguna otra indicación que requiera una explicación sobre el formulario es importante posicionar esa información **antes del propio formulario**, de forma que todos los usuarios localicen dicha información antes de rellenar el mismo.

No obstante, si esta información es muy extensa es posible incluir en el formulario una leyenda.

En el caso de que se desee incluir información sobre algún campo del formulario mediante el uso de una **leyenda**, se debe realizar de la siguiente forma:

- Si el texto de la leyenda no es muy extenso (por ejemplo, el formato de una fecha), se puede incluir inmediatamente después del texto de la etiqueta.
- Si la leyenda es muy amplia, se podría situar después del formulario y hacer referencias desde el campo correspondiente mediante enlaces ancla a dicha leyenda. Finalmente se deberá incluir un enlace de retorno al campo de origen para que el usuario pueda seguir rellenando el formulario en el orden adecuado.

4.10. SUGERENCIAS ANTE ERRORES

Cuando la información proporcionada por el usuario es incorrecta, pero se conoce un posible valor correcto entonces es recomendable sugerir un texto con la corrección. De esta forma se facilita a los usuarios la corrección de los errores.

En ese caso, las sugerencias o los enlaces a las mismas se deben situar de forma que sean fácilmente localizables por el usuario, cerca del campo de formulario

donde se ha producido el error. Por ejemplo, al comienzo del formulario o justo antes o después del campo de formulario.

Algunos ejemplos de sugerencias pueden ser:

- Correcciones ortográficas.
- Valor similar dentro de un conjunto de valores posibles. Por ejemplo, nombres de ciudades o provincias similares al introducido por el usuario.
- Preguntas adicionales para refinar datos ambiguos. Por ejemplo, "Madrid" puede referirse a la ciudad o a la provincia.
- Alternativas similares para evitar repetición de valores. Por ejemplo, durante un registro si el nombre de usuario ya existe se pueden sugerir otros similares que estén libres.

4.11. PREVENCIÓN DE ERRORES (LEGALES, FINANCIEROS, PÉRDIDA DE DATOS)

Si un error a la hora de realizar un proceso online ya es de por sí un problema que ha de evitarse, es aún más importante cuando dicho error puede tener importantes implicaciones legales o económicas para los usuarios:

- Pagos y devoluciones con carácter legal como el pago de impuestos, tasas, multas, devoluciones fiscales, etc.
- Transacciones económicas en general.
- Modificación o borrado de datos del usuario, como cuentas o perfiles de usuario, información personal, información legal, documentos personales, etc.

Si este tipo de transacciones tienen lugar inmediatamente y no se pueden deshacer pueden tener importantes y costosas consecuencias. Por ejemplo:

- Introducir mal la cantidad a pagar en un impuesto.
- Introducir un número de cuenta erróneo para una devolución.
- Cometer un error al introducir el NIF para una solicitud de una prestación.
- Etc.

Lo ideal es que dichas acciones no se llevasen a cabo inmediatamente y que los usuarios dispusiesen de un tiempo para poder cancelarlas. Por ejemplo, si el usuario borra algún documento personal que éste siga disponible en una especie de *"papelera de reciclaje"* o si realiza una transacción o pago que disponga de un tiempo durante el cual pueda cancelar o modificar la acción realizada. En este caso, se tendría que informar del tiempo disponible y de las vías disponibles para la cancelación, que no tendría por qué ser online, sino que se podrían emplear otras vías como el correo electrónico, teléfono, etc.

Sin embargo, según la naturaleza de la acción a realizar, en muchas ocasiones no es posible hacer que las acciones sean reversibles. Por tanto, en una transacción de este tipo que tenga lugar inmediatamente y no se pueda deshacer, se debe dar al menos la posibilidad de revisar la información antes de enviarla para que los usuarios puedan confirmar los datos o corregir los posibles errores detectados.

Una técnica para lograr este objetivo es proporcionar una casilla de verificación para la confirmación de los datos. De esta forma se consigue que los usuarios deban marcar la casilla de verificación para indicar que han revisado los datos y que estos puedan ser enviados. Esto es especialmente importante cuando los cambios no se pueden deshacer o cuando implican el borrado de datos.

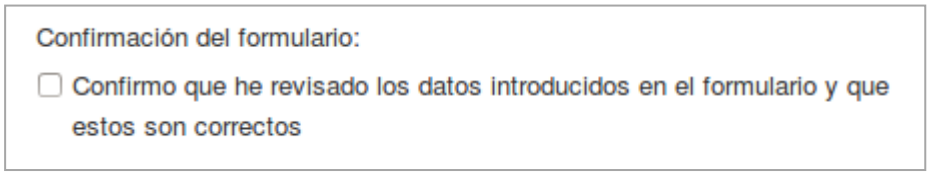

Figura 13. Casilla de verificación para confirmación del formulario

La casilla de verificación debe estar situada cerca del botón de envío para que se vea fácilmente y no tiene que estar seleccionada inicialmente, obligando al usuario a seleccionarla para poder enviar el formulario.

Otra opción es solicitar una confirmación adicional por parte del usuario después de dar al botón de envío, informando de que la acción a realizar es inmediata e irreversible, así como de sus posibles implicaciones, y dando las opciones de continuar (confirmar, enviar, etc.) y cancelar.

A la hora de revisar y corregir los datos, los usuarios podrán hacerlo directamente en los formularios de un solo paso donde todos los datos introducidos son visibles en una misma página. En cambio, en formularios donde se introducen datos en varios pasos hay dos opciones para que los usuarios puedan revisar y corregir dichos datos:

- Permitir a los usuarios retroceder a los pasos anteriores del formulario para ver y/o modificar los datos introducidos. Para cumplir esto los datos ya introducidos no podrían perderse al moverse por los pasos del proceso.

- Al terminar el último paso, proporcionar un resumen con todos los datos antes de finalizar, solicitando al usuario que los revise y permitiéndole corregir los errores si fuese necesario.

4.12. INDEPENDENCIA DE DISPOSITIVO

El acceso mediante teclado es el principal medio de interacción de las páginas accesibles ya que una gran parte de los productos de apoyo se basan en el uso del teclado o emulan su funcionamiento.

Si toda la funcionalidad es operable mediante teclado, entonces será accesible tanto para los usuarios de teclado como para los usuarios de productos de apoyo que simulan entradas de teclado como medio de interacción o que emplean el teclado de forma adicional.

4.12.1. Accesible por teclado

Es necesario asegurarse de que todos los elementos de interacción, incluidos los formularios, pueden ser seleccionados y activados a través del teclado o mediante productos de apoyo.

4.12.1.1. Uso de controles estándar

En el caso de los formularios, se deben usar de forma adecuada los enlaces y controles estándar de HTML:

- Las aplicaciones de usuario proporcionan acceso desde teclado para los enlaces y controles de formulario estándar.

- Las aplicaciones de usuario y productos de apoyo podrán reconocer la función de los controles estándar y transmitírsela a los usuarios.

4.12.1.2. Si se emplean scripts en los formularios

En caso de usar scripts y elementos de programación se deben emplear manejadores de evento que puedan lanzarse mediante teclado. De esta forma se permite que las personas que dependen del teclado puedan acceder a la funcionalidad del contenido.

Para ello, se ha de procurar usar manejadores de evento lógicos (que no dependan de dispositivos concretos) como `onfocus`, `onblur`, `onselect`, `onload`, `onsubmit` y `onreset`. Estos manejadores reconocen eventos tanto del ratón como del teclado.

Si es necesario usar manejadores de evento específicos de ratón se debe proporcionar también el manejador de evento de teclado equivalente para ejecutar la misma función. Por ejemplo:

- `onmousedown` con `onkeydown`
- `onmouseup` con `onkeyup`
- `onmouseover` con `onfocus`
- `onmouseout` con `onblur`

Aunque `onclick` sea, en principio, un manejador de evento de ratón la mayoría de los navegadores HTML lo interpretan como el manejador de evento de acción por defecto para enlaces y botones y se activa tanto con el ratón como con el teclado.

Por tanto, para documentos HTML, se puede usar el manejador de evento `onclick` sin necesidad de duplicarlo con `onkeypress`, evitando así posibles problemas con la tabulación.

Para asegurar que las acciones y eventos que lanzan funciones de scripts se pueden invocar desde el teclado es necesario asociarlas a elementos HTML accionables por teclado de forma nativa: enlaces y botones. Si se añaden funciones de script, aunque sea con eventos de teclado, a elementos que no son de interacción (como elementos `SPAN`, `DIV`, etc.) estos no serán accesibles mediante teclado al no recibir el foco y los productos de apoyo no reconocerán su función.

Finalmente, hay que asegurar que la navegación mediante el tabulador no queda atrapada en parte del contenido de la página. Se considera que el foco queda atrapado en parte del contenido si el foco entra normalmente con el teclado, pero, una vez dentro, los usuarios no pueden o no saben cómo sacarlo usando únicamente el teclado. Esto no impide que el foco quede restringido a una zona del contenido (p. ej. el *widget* de un calendario para seleccionar una fecha) mientras se opera con él siempre que sea posible recuperar el foco con el teclado de forma normal o se informa previamente sobre cómo hacerlo.

No debemos olvidarnos de las personas que emplean magnificadores de pantalla. Para este tipo de usuarios que sólo pueden ver una parte pequeña de la página es importante que la interacción sea correcta. Así, cuando se muestre un contenido adicional como reacción al hover del ratón o al foco del teclado (p. ej. un menú desplegable que tape parte de la página) entonces el usuario debe poder ocultar el contenido si así lo desea sin necesidad de mover el ratón o el foco, o bien ha de poder interactuar adecuadamente con el contenido desplegado. Este aspecto se trata más en detalle en el apartado 5.7.2 "Contenido con hover o focus".

4.12.2. Orden de tabulación

Hay que asegurar que, además de un orden de lectura correcto, el orden del foco por los elementos de interacción también tiene sentido.

Para ello se pueden situar los elementos de interacción en el código de forma que el orden de tabulación por defecto siga un orden lógico, que tenga sentido.

Si por alguna circunstancia es necesario cambiar el orden del foco por defecto, entonces hay 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 usará el atributo `tabindex` en aquellos elementos de interacción para los que se quiera modificar su orden, asegurándose siempre que el orden sigue teniendo sentido.

4.12.3. Foco visible

Si el foco es visible facilita el uso de las páginas web mediante el teclado al permitir reconocer visualmente y en todo momento cuál es el componente con el cual se está interactuando. Asimismo, resaltar los elementos cuando reciben el foco también sirve para que los usuarios sepan que se trata de elementos de interacción.

La forma más directa y sencilla de cumplir este requisito es usando los controles estándar de HTML ya que los navegadores resaltan los controles y elementos de interacción estándar cuando reciben el foco. Generalmente, los suelen resaltar mediante un borde punteado o, en campos de introducción de texto, mediante el cursor de edición (línea vertical parpadeando).

Por tanto, para asegurar que el indicador del foco del teclado permanece visible hay que usar el indicador de foco por defecto del sistema, sin modificarlo. En caso de que se quiera modificar la presentación del indicador del foco (p. ej.

mediante CSS), se hará únicamente para mejorar la visibilidad que tiene por defecto, no para eliminarlo.

4.13. PREDICTIBILIDAD ANTE CAMBIOS DE CONTEXTO

El contenido de las páginas y la funcionalidad de los formularios y diferentes componentes interactivos y funcionales deben ser predecibles para los usuarios.

Por tanto, es necesario **evitar que se produzcan cambios de contexto** cuando los usuarios no esperan que se produzcan. Se entiende por cambio de contexto a un cambio importante en el contenido de la página que, si se realiza sin avisar, puede desorientar a los usuarios.

- Cambio de la aplicación que está usando el usuario como, por ejemplo, abrir un cliente de correo electrónico, un visor de documentos PDF, etc.
- Cambio de la pestaña o ventana del navegador.
- Apertura de una nueva página o cambio significativo del contenido de la página de forma que se altera el significado principal de la misma.

Los cambios de contexto han de producirse únicamente como respuesta a una acción de los usuarios. Se consideran acciones el pulsar un botón o seguir un enlace.

El resto de las acciones no pueden generar un cambio de contexto. Por ejemplo:

- Activar o desactivar una casilla de selección (checkbox).
- Escribir en un campo de texto.
- Seleccionar una opción en un menú de selección (select).
- Etc.

Es decir, no se pueden provocar cambios de contexto cuando se cambie el estado o valor de un control de formulario, o cuando un control reciba el foco del teclado, a no ser que previamente se avise a los usuarios sobre este comportamiento. En este caso, la forma de avisar a los usuarios ha de ser mediante un mensaje de texto situado antes del control o al comienzo de formulario informando sobre dicho comportamiento.

5. OTROS REQUISITOS DE ACCESIBILIDAD PARA SEDES ELECTRÓNICAS

Los aspectos más específicos de accesibilidad con respecto a las sedes electrónicas los constituyen el uso de formularios electrónicos y el uso de certificados digitales y firma electrónica. No obstante, el resto del contenido Web incluido en dichas Sedes (tablas de datos, listas, encabezados, imágenes, objetos programados, etc.) deben cumplir todos los requisitos de accesibilidad generales al igual que si estuvieran presentes en cualquier otro sitio web.

En los siguientes apartados se expondrán algunas **cuestiones generales de accesibilidad** (las más frecuentes en sedes) que también deben ser tenidas en cuenta por los técnicos responsables del desarrollo de una Sede electrónica.

5.1. INCLUSIÓN DE TABLAS DE DATOS

En caso de que se incluyan tablas, éstas deben ser utilizadas para mostrar información tabular (datos relacionados y estructurados) y no para dotar de presentación a los contenidos.

Uno de los requisitos principales de una tabla de datos es que cada **celda de encabezado** (cabecera) se identifique mediante el elemento `TH`.

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** correspondientes, 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 (cabeceras) relacionados con ésta.

Por otro lado, para mejorar la comprensión de la estructura y de los contenidos de una tabla, es muy recomendable incluir un **título** que describa brevemente la naturaleza de la tabla. Al proporcionar un título para la tabla, se deberá hacer por medio del elemento `CAPTION`.

Asimismo, en las tablas de datos complejas es necesario incluir un **resumen** de los contenidos de esta y de las relaciones entre las celdas, especialmente para las tablas con encabezamientos anidados, celdas que ocupan varias columnas o filas, u otras relaciones que solo son evidentes cuando la tabla se muestra visualmente. En caso de ser necesario el resumen, éste se ha de incluir por medio del atributo *`summary`*.

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 no se debería emplear el atributo *summary* para incluir información de resumen. Dicha información se puede incluir de forma alternativa como un párrafo previo a la tabla.

En caso de que en una tabla de datos se proporcione un título y un resumen entonces el contenido de ambos debe ser diferente y complementario y por tanto no estar duplicado.

Ejemplo de código 17

```
<table summary="Relación de expedientes tributarios de Nombre Apellido1 Apellido2 (NIF) durante los años 2007, 2008 y 2009, en la que para cada expediente se presenta el tipo de procedimiento, el estado de la tramitación, la fecha de la última tramitación y el número de expediente">

<caption>Expedientes tributarios de Nombre Apellido1 Apellido2 (NIF)</caption>
<thead>
  <tr>
    <td></td>
    <th id="proc">Procedimiento</th>
    <th id="est">Estado de tramitación</th>
    <th id="fec">Fecha última actuación</th>
    <th id="exp">Expediente</th>
  </tr>
</thead>
<tbody>
  <tr>
    <th id="anio2007">2007</th>
    <td headers="proc anio2007">Borrador de declaración de IRPF</td>
    <td headers="est anio2007">Finalizado</td>
    <td headers="fec anio2007">15/04/2008</td>
    <td headers="exp anio2007">01010101010101OA</td>
  </tr>
  <tr>
    <th id="anio2008">2008</th>
    <td headers="proc anio2008">Impuesto sobre la Renta de las Personas Físicas. Declaración y documento de ingreso o devolución. (Mod. 100)</td>
    <td headers="est anio2008">Finalizado</td>
    <td headers="fec anio2008">27/05/2009</td>
    <td headers="exp anio2008">01010101010101OB</td>
  </tr>
  <tr>
    <th id="anio2009">2009</th>
    <td headers="proc anio2009">Impuesto sobre la Renta de las Personas Físicas. Declaración y documento de ingreso o devolución. (Mod. 100)</td>
```

```
    <td headers="est anio2009">En tramitación</td>
    <td headers="fec anio2009">07/06/2010</td>
    <td headers="exp anio2009">010101010101010C</td>
  </tr>
</tbody>
</table>
```

5.2. IDENTIFICACIÓN DE LISTAS

Cualquier enumeración de elementos que guarden algún tipo de relación entre sí debe ser identificada como **lista** mediante el marcado (X)HTML correspondiente:

- Las **listas no numeradas** se identifican mediante el elemento `<ul>`, mientras que sus elementos se definen mediante la etiqueta `<li>`.

Ejemplo de código 18

```
<h1>Oficina Virtual</h1>

<ul>
 <li>Sede Electrónica</li>
 <li>Fondo Estatal para el Empleo y la Sostenibilidad Local</li>
 <li>Fondo Estatal Inversión Local</li>
 <li>Procesos selectivos de empleo público</li>
 <li>Extranjería</li>
 <li>Canarias: Compensación transporte mercancías</li>
</ul>
```

- Las **listas numeradas** se identifican mediante el elemento `<ol>`, mientras que sus elementos se definen mediante la etiqueta `<li>`.

Ejemplo de código 19

```
<ol>
 <li>Resoluciones 11 enero</li>
 <li>Resoluciones 15 enero</li>
 <li>Resoluciones 22 enero</li>
 <li>Resoluciones 25 enero</li>
 <li>Resoluciones 29 enero</li>
</ol>
```

- Las **listas de definición** se identifican mediante las etiquetas `<dl>`, `<dt>` y `<dd>`, de forma que la etiqueta `<dl>` crea la lista de definición y las etiquetas `<dt>` y `<dd>` definen respectivamente el término y la descripción o definición de cada elemento de la lista.

Ejemplo de código 20

```
<dl>
 <dt>Delegación:</dt>
   <dd>Comunidad Autónoma de Andalucía</dd>
 <dt>Dirección:</dt>
   <dd>Plaza de España – Torre Sur, 41071 Sevilla</dd>
 <dt>Teléfono:</dt>
   <dd>95 556 90 00</dd>
 <dt>Fax:</dt>
   <dd>95 423 20 77</dd>
</dl>
```

El marcado de las listas debe tener una estructura correcta y, en ningún caso, deben ser simuladas mediante elementos que no han sido creados para tal fin (por ejemplo, párrafos iniciados con asterisco, guiones o números).

5.3. DEFINICIÓN DE ENCABEZADOS O TÍTULOS DE PÁGINAS

Es necesario definir la estructura de las páginas a través de **elementos de encabezado o título**, con diferentes niveles de profundidad, permitiendo acceder rápidamente a las secciones de éstas. Para ello se deben utilizar los elementos de (X)HTML adecuados (`H1`-`H6`), y no características de presentación (por ejemplo, efectos de fuente).

Además, esta estructura de encabezados debe seguir una anidación correcta en la que no se produzcan saltos de nivel, es decir, no se debe pasar directamente de un encabezado de nivel 1 a otro de nivel 3, sino que debe haber un nivel 2 intermedio.

Ejemplo de código 21

```
<h1>Sede Electrónica del Ministerio de la Presidencia</h1>
<p>Bienvenido a la sede electrónica del Ministerio de la Presiden-
cia, está usted en la página de Inicio</p>
<h2>Información de la sede</h2>
<ul>
  <li>Normativa reguladora</li>
  <li>Información sobre el Registro de la Sede</li>
  <li>Verificación de certificado</li>
  <li>Hora oficial de la sede</li>
  <li>Cartas de servicio</li>
  <li>Canales de acceso</li>
  <li>Calendario de días hábiles</li>
  <li>Quejas y sugerencias</li>
  <li>Firma electrónica</li>
  <li>Notificaciones electrónicas</li>
</ul>
<h2>Órganos y organismos de la sede</h2>
<ul>
```

```
  <li>Dirección General De Coordinación De La Administración peri-
férica Del Estado</li>
  <li>Secretaría General Técnica</li>
  <li>Dirección General De Recursos Humanos, Servicios E Infraes-
tructura</li>
  <li>Subsecretaría De La Presidencia</li>
</ul>
<h2>Subsedes</h2>
<ul>
  <li>Centro de Estudios Políticos y Constitucionales</li>
  <li>Centro de investigaciones Sociológicas</li>
</ul>
<h2>Otras sedes</h2>
<ul>
  <li>BOE</li>
  <li>CSD</li>
</ul>
```

En caso de que la gramática empleada sea HTML 5 se pueden emplear los nuevos elementos para la creación de secciones dentro de un documento.

Tabla 2. Nuevos elementos para la creación de secciones en HTML 5

ELEMENTO	FUNCIÓN
<section>	Delimitador genérico de secciones de contenido de una página, como capítulos, apartados en un artículo, etc.
<article>	Representa un contenido que es autocontenido y que se puede redistribuir o reusar de forma independiente (por ejemplo, mediante sindicación).
<aside>	Representa una digresión o ruptura respecto al hilo o discurso del contenido que le rodea, con el que está relacionado tangencialmente, pero del que debería considerarse como un contenido aparte.
<nav>	Representa una sección con enlaces de navegación a otras páginas o a otras partes dentro de la misma página.

En HTML5 los elementos `H1` a `H6` representan encabezados de secciones de contenido. No existen diferencias respecto a versiones anteriores de HTML. Se considera que el encabezado `H1` es el de mayor rango y el encabezado `H6` el de menor rango.

Al igual que ocurre en HTML 4, los elementos `H1- H6` crean nuevas secciones de contenido de forma implícita. A partir de estas secciones se forma el *outline* o mapa de contenidos de la página que refleja su estructura. Los usuarios pueden usar el mapa de contenidos para visualizar su estructura o navegar por la página (p. ej. los usuarios de lectores de pantalla)

Las secciones de una página se pueden crear de forma implícita mediante los elementos de encabezado habituales o de forma explícita mediante los elementos de sección de contenido (`SECTION`, `ARTICLE`, `ASIDE`, `NAV`). Estas secciones también se usan para definir el *outline* o mapa de contenidos del documento.

Como novedad en HTML 5, el primer encabezado de una sección creada con alguno de estos elementos puede ser de nivel `H1`, independientemente del nivel del último encabezado usado anteriormente. Esto facilita la contribución de contenidos de terceras partes, o la inclusión de contenidos mediante gestores de contenido, porque no es necesario preocuparse del anidamiento de los encabezados siempre y cuando dichos contenidos se incluyan dentro de una sección creada explícitamente con alguno de estos elementos (`SECTION`, `ARTICLE`, `ASIDE`, `NAV`).

En la especificación de HTML5 se puede consultar información detallada sobre el uso de encabezados, secciones y generación del *outline* del documento.

5.4. INCLUSIÓN DE IMÁGENES

Para toda imagen, ya sea informativa, funcional, textual, decorativa o compleja, se debe proporcionar una **alternativa textual** que aporte la misma información o función que la imagen.

Este texto alternativo deberá ser:

- Descriptivo de la información o función de la imagen.
- No demasiado largo.
- Preferentemente sin abreviaturas.

Ejemplo de código 22

```
<img alt="Sede Electrónica del Ministerio de la Presidencia"
src="../logo_sede.jpg" />
```

Las imágenes simplemente decorativas deberán tener un texto alternativo vacío (*alt*="") sin espacio entre las comillas.

Ejemplo de código 23

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

En el caso de **imágenes complejas** a través de las cuales se transmite mucha información (gráficas, diagramas, mapas, etc.), además de ofrecer una alternativa textual que identifique brevemente el tipo de información transmitida por la imagen, es necesario proporcionar una descripción detallada en una página aparte o en la misma página en la que se encuentra la imagen.

En HTML hasta la versión 4.01 y XHTML la URL de esta descripción detallada se indicará en el atributo *longdesc* 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.

Ejemplo de código 24

```
<img alt="Esquema de organismos de la sede" long-
desc="https://sede.gob.es/descripcion.html" src="../Organismos-
Sede.jpg" />
<a href="https://sede.gob.es/descripcion.html">Descripción deta-
llada de la imagen</a>
```

En cambio, en HTML5 el atributo *longdesc* pasa a considerarse como obsoleto. 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. Para ampliar información sobre las descripciones detalladas sin emplear el atributo *longdesc*, se puede consultar las suficientes técnicas para las descripciones largas en imágenes complejas.

Por otra parte, es necesario recordar que las imágenes incluidas deben poseer un contraste suficiente entre los colores de fondo y primer plano. Es necesario asegurar un contraste suficiente tanto en el texto mostrado en las imágenes como en aquellos contenidos gráficos cuya información visual es necesaria para su identificación o comprensión como pueden ser los iconos, diagramas, gráficas, infografías, etc.

En las sedes electrónicas también puede haber presente contenido multimedia. Este aspecto se trata en la ***"Guía de accesibilidad en contenidos multimedia"*** disponible en el área de documentación del Portal de la Administración Electrónica (PAe).

5.5. IDENTIFICACIÓN DEL IDIOMA PRINCIPAL Y LOS CAMBIOS DE IDIOMA

Los lectores de pantalla pueden usar el acento, entonación y pronunciación adecuada para cada idioma. Si un lector de pantalla lee un texto con una pronunciación que no es la adecuada para el idioma es posible que dicho texto resulte incomprensible. Para evitar este problema se debe identificar tanto el idioma principal usado en la página como los cambios de idioma que se produzcan en el contenido de esta.

El idioma de la página se identifica en el elemento `HTML` mediante los atributos *`lang`* y/o *`xml:lang`* según las siguientes reglas:

- 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 y 1.1 servido como xml: atributo **xml:lang**.

Como se ha comentado antes, si se utilizan varios idiomas en una misma página se debe asegurar que cualquier cambio de idioma esté indicado. El marcado de estos cambios de idioma se realizará con los atributos *`lang`* y/o *`xml:lang`* según las mismas reglas que para el idioma principal.

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 `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`.

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.

Un caso típico en el marcado de cambios de idioma lo constituyen los **menús de selección de idioma**, en los que se incluyen enlaces a las versiones traducidas

de un sitio web, que están escritos en un idioma distinto al idioma principal de la página en la que se encuentran.

Figura 14. Menú de selección de idioma

Ejemplo de código 25

```
<ul>
 <li>Bienvenido</li>
  <li><a href="/cambioidioma/ca/" hreflang="ca" xml:lang="ca"
        lang="ca" title="Canviar a Català">Benvingut</a></li>
 <li><a href="/cambioidioma/gl/" hreflang="gl" xml:lang="gl"
        lang="gl" title="Cambiar a Galego">Benvido</a></li>
 <li><a href="/cambioidioma/eu/" hreflang="eu" xml:lang="eu"
        lang="eu" title="-ra aldatu Euskara">Ongi Etorri</a></li>
 <li><a href="/cambioidioma/en/" hreflang="en" xml:lang="en"
        lang="en" title="Change to English">Welcome</a></li>
 <li><a href="/cambioidioma/fr/" hreflang="fr" xml:lang="fr"
        lang="fr" title="Changer Français">Bienvenue</a></li>
</ul>
```

En el caso de que la implementación del menú de selección de cambio de idioma se realice mediante la representación gráfica de las banderas correspondientes a ese territorio o mediante siglas técnicas de codificación del idioma (es, eu, ca, etc.) se debe especificar sin lugar a duda la relación entre bandera o sigla con un determinado idioma.

5.6. ENLACES E INTERACCIÓN

5.6.1. Textos descriptivos

Los enlaces que se incluyan en cualquier punto de la sede deben poseer un **texto suficientemente descriptivo** para comprender su finalidad o destino. Idealmente deben ser significativos cuando se lean fuera de su contexto, es decir, leyendo únicamente el texto del enlace. Sin embargo, como caso especial, se admite que los enlaces sean significativos a partir de su contexto más inmediato. Se considera contexto inmediato aquel al que pueden acceder los lectores de pantalla a partir del enlace cuando tabulan por la página, como la frase, el párrafo, el elemento de lista o la celda de tabla que contiene al enlace, o el encabezado de la sección en la que se encuentra.

Por otro lado, se recomienda evitar la **apertura de enlaces en nuevas ventanas** del navegador. En caso de que dicha apertura resulte completamente necesaria, se recomienda informar de la misma.

Concretamente, para las Administraciones Públicas y en particular para las sedes electrónicas, se considera como necesaria o recomendable la apertura de ventana en los siguientes casos:

- Enlaces a sitios web diferentes al que nos encontremos en ese momento: sitios web externos y servidores específicos de aplicación.
- Enlaces a archivos adjuntos (pdf, txt, xml, etc.).

Además, para el caso de las sedes electrónicas es obligatorio informar al usuario del momento en el que se abandona la sede electrónica.

5.6.2. Aviso de apertura en nueva ventana

En la norma UNE-EN 301549 avisar de la apertura de nuevas ventanas o pestañas del navegador no es obligatorio, aunque sigue siendo una buena práctica y es muy recomendable su uso ya que mejora tanto la accesibilidad como la usabilidad de la página.

Si avisamos por medio del atributo *title* del enlace los usuarios de teclado nunca leerán la información que proporcionamos en el título del enlace. Por lo que se recomienda encarecidamente no utilizar el atributo *title* como técnica para hacer accesible un sitio web.

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 26

```
<a href="recpet.html" target="_blank">Recursos y Peticiones (se
abre en ventana nueva)</a>
```

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

Ejemplo de código 27

```
<a href="nueva_ventana.html" target="_blank">Recursos y Peticiones
<img alt="Se abre en ventana nueva" src="icono.gif" /></a>
```

Para los **enlaces gráficos**:

- Incluyendo el aviso de nueva ventana en la alternativa textual de la imagen:

Ejemplo de código 28

```
<a href="http://www.060.es" target="_blank">
   <img src="logo_060.jpg"
   alt="060, Portal del Ciudadano. 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.

Estas mismas técnicas son aplicables si se debe informar de la salida de la sede electrónica. En ese caso los mensajes a incluir podrían ser *"Se abre en ventana nueva y se abandona la sede electrónica"*.

5.7. INTERACCIÓN

5.7.1. Atajos de teclado

En el caso de que para activar los elementos de interacción (enlaces, botones, etc.) se empleen **atajos de teclado usando una única letra**, signo de puntuación, número o símbolo entonces se debe cumplir al menos una de las siguientes condiciones:

- Existe un mecanismo que permite **desactivar** el atajo de teclado
- Existe un mecanismo que permite **reasignar** el atajo de teclado para emplear en su lugar otra tecla no imprimible (ej, *Ctrl*, *Alt*, etc.)
- El atajo de teclado **sólo se puede activar cuando el componente tiene el foco** del teclado.

El objetivo es evitar que las personas que interactúan mediante entradas de voz activen los elementos de interacción de forma accidental y no intencionada cuando pronuncien la letra correspondiente por otros motivos diferentes.

Esto no aplica a las teclas de acceso rápido (`accesskey`) sino a los atajos de teclado personalizados y programados de forma específica para emplear un único carácter para su activación. Las teclas de acceso rápido requieren de la pulsación de una tecla modificadora para su activación (*Ctrl*, *Shift*, *Alt*, etc.) de forma que no se activan accidentalmente al emplear entrada por voz.

5.7.2. Contenido con hover o focus

Si cuando un elemento recibe o pierde el puntero (p. ej. `hover` del ratón) o el foco del teclado (`focus`) se muestra u oculta algún contenido adicional, entonces se debe cumplir lo siguiente:

- Debe haber disponible un mecanismo que permita **descartar el contenido adicional** sin necesidad de mover el puntero o cambiar el foco del teclado a no ser que el contenido adicional informe acerca de un error en la entrada de datos o no tape o reemplace otro contenido.

- Si el puntero puede mostrar el contenido adicional, entonces el **puntero se puede mover sobre el contenido adicional** sin que éste desaparezca.

- **El contenido adicional debe permanecer visible** hasta que se retire el puntero o el foco del teclado o bien el usuario lo descarte (por el primer punto) o su información no siga siendo válida.

Este requisito se entiende mejor con ejemplos. Hace referencia a los contenidos que se muestran u ocultan con la interacción del usuario al situar el puntero sobre algunos elementos o cuando reciben el foco del teclado como es el caso de menús desplegables, pop-ups no modales, tooltips, etc. El objetivo es asegurar que los usuarios puedan tanto percibir el contenido mostrado como descartarlo si no lo requieren y sin que perjudique la experiencia de uso. La razón para esto es porque ciertos usuarios, como aquellos que emplean magnificadores de pantalla, sólo perciben visualmente un fragmento de la página y no todo su contenido de forma simultánea y se pueden encontrar con situaciones no deseadas como:

- Haber mostrado el contenido adicional de forma involuntaria. Por ejemplo, al pasar el puntero sobre un menú desplegable mientras lo desplazaba hacia otra zona de la página, sin querer mostrar el menú.

- No percatarse de que el contenido adicional se ha mostrado. Por ejemplo, si el contenido mostrado está fuera de su rango de visión y fuera del alcance del puntero de ratón.

- El contenido adicional interfiere con la habilidad del usuario para realizar una tarea, si se muestra tapando u ocultando otro contenido al que necesite acceder.

Se consideran como **excepciones** aquellos contenidos que se muestran u ocultan cuya presentación visual está controlada por la aplicación de usuario y no por los desarrolladores como, por ejemplo, el tooltip que muestra el contenido de los atributos `title` (título).

5.7.3. Cancelación del puntero

Toda funcionalidad que se pueda operar mediante un puntero sencillo (*single pointer*) debe cumplir al menos una de las siguientes condiciones:

- El **evento *down* del puntero no se emplea** para ejecutar ninguna parte de la funcionalidad.
- La función se completa con el evento *up* del puntero y existe un **mecanismo para cancelar la función** antes de que se complete **o para deshacerla** una vez completada.
- El **evento *up* del puntero deshace** cualquier consecuencia del evento *down* previo.
- Completar la función con el **evento *down* es esencial** (p. ej. se considera esencial un emulador de teclado)

Este requisito se refiere a los métodos de interacción con puntero, como puede ser el empleo del ratón o bien, en una pantalla táctil, la interacción con el dedo.

El objetivo es prevenir interacciones accidentales por parte de los usuarios al pulsar sobre algún componente sin querer. Por ejemplo, personas con problemas de movilidad (falta de precisión, temblores, etc.) pueden pulsar o tocar la pantalla sin querer con resultados no deseados. La mejor forma de evitar esto es lanzar las acciones con el evento *up* en vez de con el evento *down*.

El evento *up* sólo se dispara cuando se levanta el dedo o se libera el clic del ratón dentro de los límites del elemento de interacción. Así, si los usuarios se dan cuenta de que han pulsado un botón o enlace sin querer (al cambiar el indicador visual del foco, por ejemplo) aún tienen la posibilidad de deslizar y levantar el puntero fuera del área de interacción, cancelando así la acción.

5.7.4. Mensajes de estado

Los **mensajes de estado** deben poder ser **identificados por software a través de sus roles o propiedades** de forma que se puedan transmitir a los usuarios sin necesidad de recibir el foco. Es decir, que los lectores de pantalla y los productos de apoyo similares puedan identificar y reconocer automáticamente estos mensajes de estado para que así se los puedan transmitir a los usuarios automáticamente sin que éstos tengan que desplazarse hasta su ubicación para leer su contenido. El objetivo es que los usuarios sean conscientes de cualquier cambio relevante en el contenido, aunque no tenga el foco en ese momento.

Se entiende por un **mensaje de estado** aquellos mensajes que proporcionan información al usuario acerca del éxito o el resultado de una acción, del estado de espera de una aplicación, del progreso de un proceso o de la existencia de errores y dichos mensajes se proporcionan dinámicamente sin que se produzca un cambio de contexto (p. ej. se muestran en la misma página y no en una página nueva).

No toda la información que se añade en la página entra dentro de la definición de un mensaje de estado. Por ejemplo, en los resultados de una búsqueda el propio listado con los resultados no son un mensaje de estado. Sin embargo, un texto que informe acerca de la situación de la búsqueda (*"buscando…"*) o acerca del número de resultados obtenidos (*"9 resultados obtenidos"*, "*Sin resultados*", etc.) sí se considera un mensaje de estado si éste no recibe el foco.

Para hacer que los mensajes de estado sean reproducidos por los lectores de pantalla hay que emplear **roles de WAI-ARIA** para identificar la finalidad del componente. Algunos roles tienen la característica de que los lectores de pantalla leen su contenido automáticamente cuando éste cambia o cuando se muestra en la página. Así, dependiendo de la situación podemos emplear diferentes tipos de roles:

- Si se trata de un mensaje de estado avisando del **éxito o del resultado de una acción**, con el atributo `role="status"` con un texto informando acerca del envío correcto o del resultado obtenido.

Ejemplo de código 29

```
<div role="status">5 resultados obtenidos.</div>
```

Ejemplo de código 30

```
<p role="status"><img src="carrito.png" alt="Carrito de la com-
pra"><span id="cart">0</span> elementos</p>
```

- Si se trata de una sugerencia o **aviso acerca de la existencia de un error**, con el atributo *role="alert"* junto con alguna técnica de las empleadas para informar acerca de errores en la entrada de datos (identificar los campos obligatorios no completados, informar acerca de los errores en el formato o valor de los datos, dar sugerencias de corrección del error, etc.

Ejemplo de código 31

```
<p id="errors" role="alert">
  El campo email es obligatorio.
</p>
```

- Si se trata de un mensaje de estado acerca del **progreso en un proceso**, con *role="progressbar"*, con *role="log"* (para identificar actualizaciones secuenciales de información) o con *role="status"* y un texto explicativo.

Ejemplo de código 32

```
<div role="progressbar" aria-valuenow="20" aria-valuemin="0" aria-
valuemax="100">20 %</div>
```

Ejemplo de código 33

```
<div id="chatRegion" role="log">
  <h4>Chat</h4>
    <ul id="conversation">
      <li>Primer mensaje del chat</li>
      <li> ... </li>
      <li>Penúltimo mensaje del chat</li>
      <li>Último mensaje del chat</li>
    </ul>
</div>
```

Ejemplo de código 34

```
<p role="status">Se encuentra en el paso 3 de 5. Por favor, in-
cluya sus datos personales.</p>
```

Nota: cuando un contenido se marca con *role="log"* el lector de pantalla únicamente leerá las nuevas entradas que se añadan al final de este. Así, en el caso de los comentarios de un chat como el del ejemplo, el lector de pantalla leerá en voz alta el último mensaje añadido. El *role="log"* también es aplicable a un histórico de mensajes, un log de un juego o un log de errores, por ejemplo.

5.8. USO DE UNIDADES RELATIVAS

El empleo de **unidades relativas** (*em*, *rem* o *%*) 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.

Anteriormente, con las WCAG 1.0, las páginas debían adaptarse y transformarse adecuadamente sea cual sea la resolución usada y tamaño del texto. Para ello, los tamaños de fuente y bloques debían especificarse en unidades relativas en vez de unidades absolutas, pudiéndose utilizar unidades *em, rem* o *%,* en función del tipo de maquetación aplicada.

Por otra parte, 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.

Todos los **navegadores de uso común** disponen de una función de **zoom** que permite aumentar el contenido, si el contenido de la página web mantiene su legibilidad y funcionalidad entonces se considera que se cumple dicho requisito.

Sin embargo, para asegurar que se permite el redimensionado del texto en todos los navegadores, incluso en aquellos que carecen de zoom, es **necesario** seguir empleando **unidades relativas** para definir el tamaño del texto y de sus contenedores, de forma que las páginas se adapten y transformen adecuadamente con independencia del tamaño del texto.

Para que en nuestra página web se visualice correctamente en un dispositivo móvil y permita que se pueda hacer zoom con el gesto "pinch" al menos un 200% es importante definir correctamente el *viewport* de la etiqueta *META*:

Ejemplo de código 35

```
<meta name="viewport" content="width=device-width;initial-
scale=1.0; maximum-scale:2.0; user-scalable=1" />
```

Definiendo así el *viewport* estamos permitiendo que los usuarios que lo deseen puedan hacer un zoom de hasta 200%. Si el valor de *maximum-scale* es *1.0* estamos impidiendo el zoom.

Otra consideración a tener en cuenta es que **no se debe ocultar la barra de scroll horizontal** con la propiedad *overflow-x: hidden* de CSS porque puede que el usuario la necesite cuando hace zoom.

5.9. PRESENTACIÓN Y MAQUETACIÓN

Cualquier **efecto de presentación** (tipo de fuente, color del texto, espaciados, etc.) se recomienda aplicarlo a través de propiedades de **hojas de estilo CSS externas**. Con ello se evita el empleo de elementos y atributos desaconsejados y/o de presentación, los cuales ponen en riesgo la compatibilidad con cualquier tipo de agente de usuario.

Ejemplo de código desaconsejado 2

```
<u>Funcionamiento del componente de firma</u>
<b>Funcionamiento del componente de firma</b>
```

De igual modo, se desaconseja la utilización de estilos en línea (atributo *style*) y embebidos (elemento STYLE), ya que éstos dificultan el mantenimiento y la limpieza del código.

Ejemplo de código desaconsejado 3

```
<!-- Estilos CSS en línea -->

<img alt="" src="prueba.jpg" style="border-style: solid; border-
width: 5px; margin: 5px; width: 200px; height: 156px;" />

<p>Este procedimiento requiere el uso de un <span style="font-fa-
mily: comic sans ms,cursive;">certificado digital</span> recono-
cido por cualquiera de las entidades oficiales de certificación
nacionales, o del <span style="font-family: comic sans ms,cur-
sive;">DNI electrónico</span></p>

<!-- Estilos CSS embebidos -->

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

Respecto a la presentación del texto, la maquetación del sitio o aplicación web ha de realizarse de forma que **no se pierda contenido o funcionalidad cuando los usuarios modifiquen algunas de las características del texto** como el espaciado entre líneas, párrafos, letras y palabras. Por ejemplo, algunos usuarios con problemas visuales emplean hojas de estilo personalizadas para **adaptar la presentación del texto a sus necesidades**. Para facilitar el acceso al contenido a este tipo de usuarios, en las pautas de accesibilidad se exige que estos puedan:

- Ajustar el alto de la línea hasta al menos 1.5 veces el tamaño de la fuente

- Espaciar los párrafos hasta al menos el doble del tamaño de la fuente
- Ajustar el espacio entre letras hasta al menos 0.12 veces el tamaño de la fuente
- Ajustar el espaciado entre palabras hasta al menos 0.16 veces el tamaño de la fuente.

Este requisito no implica que el contenido debe tener dicha presentación inicialmente, sino que no se impida ajustarlo para obtener dicha presentación. Por ejemplo, que los usuarios puedan aplicar dichos estilos sin que se produzcan desbordamientos o solapamientos de contenidos que dificulten o impidan su visualización.

Otro aspecto que debe ser tenido en cuenta es el empleo de hojas de estilo CSS para realizar la maquetación de las páginas de las Sedes electrónicas.

En este sentido, en algunos formularios de tramitación electrónica se utilizan tablas para realizar la maquetación de los campos de acuerdo con un formato preestablecido en el que se intenta reproducir la misma maquetación que el correspondiente formulario original en papel autocopiativo.

Es importante resaltar que el **formulario electrónico no tiene por qué ser igual al de papel**, y que el tratar de reproducirlo puede generar problemas de acceso o utilización para algunos usuarios, especialmente para usuarios de lectores de pantalla para quienes resulta complicada la lectura de páginas maquetadas con tablas, más aún si se trata de formularios con un gran número de campos. En cualquier caso, para que la maquetación con tablas fuera válida, **es imprescindible que la información tenga sentido leída "linealmente"**, de izquierda a derecha y de arriba hacia abajo.

5.10. VALIDACIÓN GRAMATICAL

Resulta importante emplear un código, tanto (X)HTML como CSS, que valide correctamente respecto de la gramática utilizada, ya que ayudará a que los navegadores funcionen de manera más efectiva y a que las páginas se visualicen correctamente en la mayoría de los dispositivos.

Por otro lado, se aconseja utilizar **especificaciones estrictas** en detrimento de las transicionales, con el fin de generar páginas más robustas y garantizar la compatibilidad con futuras especificaciones.

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

- **Markup Validation Service**
- **CSS Validation Service**
- **Unicorn**

Validar que el código usado utiliza elementos y atributos estándar de HTML **no es un requisito**, aunque sí es altamente recomendable para asegurar la compatibilidad de este entre diferentes navegadores. Como requisito mínimo, el código HTML 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. Los atributos no deben estar duplicados y su valor debe estar correctamente entrecomillado. Los identificadores de un elemento asignados a través del atributo `id` deben tener un valor único e identificable con respecto al resto de identificadores de otros elementos.

5.11. GENERACIÓN DE DOCUMENTOS PDF

En las Sedes electrónicas es muy frecuente la generación de documentos PDF como acuse de recibo de solicitudes electrónicas. Estos documentos deben incluir las características de accesibilidad necesarias, que principalmente son las siguientes:

- El documento debe contener texto generado por ordenador en lugar de imágenes escaneadas.
- El documento generado debe ser un "PDF etiquetado".
- Incluir marcadores que faciliten la navegación por el documento.
- Proporcionar texto alternativo para todos los elementos no textuales.
- Definir el idioma principal del documento.

- Especificar un orden lógico de lectura.
- Especificar claramente el destino de los enlaces.
- No basar la información sólo en el color.
- Aplicar suficiente contraste al documento.
- Configurar apropiadamente la seguridad del documento.

No obstante, es aconsejable ofrecer una alternativa equivalente en otro formato ((X)HTML+CSS) que muestre al menos la información más relevante de tales documentos.

Nota: Para ampliar información sobre la construcción de documentos PDF accesibles puede consultar la "Guía de Accesibilidad en documentos PDF" disponible en el Portal de la Administración Electrónica (PAe)

5.12. LÍMITES DE TIEMPO

Las personas con discapacidad suelen necesitar más tiempo que la mayoría de los usuarios para completar determinadas tareas. Personas con discapacidades físicas, problemas visuales o usuarios de lectores de pantalla pueden necesitar más tiempo para completar los formularios o leer el contenido de las páginas.

5.12.1. Tiempos de sesión

Por cuestiones de seguridad, también se debe tener en cuenta el control del tiempo en el que está activa la sesión. Por ejemplo, para evitar que una persona olvide desconectarse de la Sede electrónica y otra aproveche su usuario cuando no se encuentre presente, pudiendo suplantar su identidad o acceder a información confidencial.

No obstante, desde el punto de vista de la accesibilidad, estos tiempos de sesión en ocasiones pueden no ser lo suficientemente apropiados para que todos los usuarios sean capaces de cumplimentar los trámites y/o consultar la información que desean sin que caduque la sesión. Este problema se puede resolver ofreciendo opciones para desactivar, ajustar o aumentar dichos límites de tiempo, de forma que no sean demasiado excesivos para mantener el nivel de seguridad y permitan que usuarios que no puedan acceder a la Sede de forma rápida dispongan del tiempo suficiente para llevar a cabo las acciones pertinentes, siendo además recomendable la inclusión en la página de un aviso del tiempo que puede llevar completar la tarea.

Se estima que el tiempo suficiente para que la gran mayoría de los usuarios pueda realizar una tarea es diez veces el tiempo medio de realización de esta. Por ejemplo, si un usuario puede rellenar un formulario en un minuto, diez minutos serían suficientes para prácticamente todos los usuarios.

También se contempla como buena práctica la posibilidad de que el usuario vuelva a autentificarse cuando el tiempo de sesión caduca, de forma que continúe con su actividad sin perder ningún dato de la página en la que se encontraba.

5.12.2. Límites de tiempo de lectura

Además de los tiempos de sesión, existen otros límites de tiempo que se suelen imponer a los usuarios, generalmente relacionados con el tiempo de lectura del contenido. Por ejemplo, un contenido el movimiento o que se desplaza está imponiendo un límite de tiempo a los usuarios para completar su lectura. De igual forma, una redirección o una actualización automática también impone un límite de tiempo para leer el contenido.

En el caso del contenido en movimiento (textos que se desplazan, imágenes o animaciones que cambian el texto mostrado en periodos regulares, etc.) hay que evitar su uso en la medida de lo posible. En caso contrario, se debe permitir a los usuarios detener dicho movimiento. Si durante el movimiento, en un momento dado, sólo se muestra un fragmento de todo el contenido entonces al detenerlo se debe mostrar el contenido en su totalidad o bien se debe permitir al usuario pausar y reanudar el movimiento para que pueda leerlo a un ritmo adecuado para sus necesidades.

Asimismo, se debe evitar el uso de actualizaciones automáticas del contenido, bien sean completas de toda la página o de una parte del contenido. En caso de usarse, se debe permitir a los usuarios detenerlas, pausarlas o controlar la frecuencia de la actualización.

En cuanto a las redirecciones automáticas, sólo se podrán usar si se realizan de forma instantánea y transparente para el usuario. Por ejemplo, usando redirecciones de servidor o bien redirecciones de cliente con un retardo de 0 segundos.

Se consideran excepciones:

- **Excepción en tiempo real**: el límite de tiempo es una parte requerida de un evento en tiempo real (por ejemplo, una subasta), y no es posible ninguna alternativa al límite de tiempo; o

- **Excepción Esencial**: El límite de tiempo es esencial y prorrogarlo invalidaría la actividad; o
- **Excepción de 20 horas**: el límite de tiempo es superior a 20 horas.

6. ACCESIBLE DESDE DISPOSITIVO MÓVIL

La accesibilidad se debe garantizar con independencia del dispositivo que el usuario utilice para acceder a los sitios web, incluyendo dispositivos móviles. En la UNE-EN 301549:2022 encontramos una serie de requisitos que afectan especialmente a los sitios web cuando se accede mediante un dispositivo móvil.

Esto no implica necesariamente que los sitios web deban ser completamente móviles, con diseños específicos y optimizados para estos dispositivos, sino que al menos se cumplan unos requisitos básicos para permitir el acceso desde este tipo de dispositivos sin que existan barreras significativas.

6.1. MAQUETACIÓN ADAPTABLE A DIFERENTES TAMAÑOS DE VENTANA

El contenido de los sitios web debe poder presentarse sin pérdida de información o funcionalidad y sin necesidad de realizar scroll en dos dimensiones en áreas de visualización reducidas (320x256 píxeles CSS). Si bien este criterio está pensado inicialmente para mejorar la experiencia de las personas con baja visión asegurando que se pueda hacer zoom hasta al menos un 400% del tamaño original sin que se produzca un doble scroll (partiendo de una resolución de 1280x1024), en la práctica e implícitamente también supone que la página se debe *"adaptar"* correctamente a diferentes tamaños de ventana.

En las WCAG 2.1 se indica que permitir el reajuste del contenido se conoce también como *Diseño Web Adaptable* (*Responsive Web Design*) y 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. Estos puntos de ruptura se disparan de igual forma tanto si se reduce el tamaño de la ventana como si se hace zoom sobre el contenido.

Por tanto, la maquetación de las plantillas empleadas en las sedes electrónicas 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** y **grid layout** de CSS o maquetación con el **modelo Flexible Box** de CSS (Flexbox).

Adicionalmente, aunque el contenido se vea correctamente en un dispositivo móvil no se debe bloquear la posibilidad de hacer zoom sobre el mismo. Se considera un error emplear, por ejemplo, elementos meta con "`maximum-scale`" o "`minimum-scale`" o bien con "`user-scalable=no`" o "`user-scalable=0`".

No obstante, es importante tener en cuenta que este requisito contempla como **excepciones aquellas partes del contenido que requieren de una presentación y maquetación en dos dimensiones** para poder transmitir su significado o para poder emplearse de forma correcta.

Dentro de esta excepción se pueden incluir las **imágenes y contenidos multimedia** ya que por naturaleza tienen dos dimensiones, aunque es posible su redimensión hasta adaptarlos al tamaño de la ventana. Una excepción más clara es el caso de las **tablas de datos**, en las que las relaciones entre las celdas se establecen en un contexto bidimensional. Por lo tanto, las tablas de datos, en especial las tablas de datos complejas, están fuera del alcance de este requisito.

Otro contenido que se puede considerar como una excepción son aquellas **interfaces de usuario complejas** como las que proporcionan barras de herramientas para editar contenidos que se deben mostrar simultáneamente junto con la barra de herramientas (editores de texto, editores gráficos, etc.). Por otra parte, aunque no se mencione de forma explícita en las WCAG 2.1, se pueden considerar también como excepción los **formularios o trámites de notoria complejidad**, tanto a nivel de interacción como de visualización, de forma que no son razonablemente operables desde dispositivos móviles.

Nota: Esta última excepción acerca de los formularios o trámites de notoria complejidad no está incluida explícitamente en las WCAG 2.1. Por tanto, desde un punto de vista estricto sí deberían poder adaptarse a dispositivos móviles. Sin embargo, desde el Observatorio de Accesibilidad se considera que existen ciertos trámites de la Administración Electrónica cuya complejidad de comprensión, operación e interacción desde dispositivos móviles iguala o supera a los casos excepcionales detallados en las WCAG 2.1 como puede ser el caso, por ejemplo, de Renta WEB (antiguo programa PADRE).

6.2. NO BLOQUEAR LA ORIENTACIÓN

El contenido **no puede restringir su visualización y funcionalidad a una única orientación de la pantalla** (horizontal o vertical) a no ser que dicha orientación específica sea esencial.

Algunos sitios web o aplicaciones están diseñados y configurados de forma que requieren que el dispositivo se use con una determinada orientación, vertical u horizontal. Sin embargo, existen usuarios que tienen los dispositivos anclados en una posición fija (p. ej. sobre una silla de ruedas) y no pueden modificar su orientación. Por ejemplo, un usuario que vaya en silla de ruedas y use una tablet anclada en el reposabrazos en posición vertical no podrá acceder a un sitio web que sólo se muestre correctamente en posición horizontal.

Por tanto, las maquetaciones de los sitios y aplicaciones web se han de realizar empleando técnicas (CSS) que no restrinjan su visualización a una determinada orientación y sin impedir o dificultar su uso en otras posibles orientaciones.

Se consideran como **excepción** a este requisito aquellos sitios o aplicaciones en los que es **esencial mostrar su contenido en una determinada orientación**. Entre los ejemplos que se documentan en las pautas de accesibilidad se incluye el caso de un cheque bancario, una aplicación de piano, una presentación para mostrarse en un proyector o televisión o un contenido de realidad virtual donde no son aplicables múltiples posibles orientaciones.

Es necesario distinguir entre la responsabilidad de los autores de contenido web respecto a no restringir la orientación del contenido y los mecanismos específicos de dichos dispositivos empleados para bloquear la orientación. Si un usuario bloquea su dispositivo en una determinada orientación se espera que todas las aplicaciones respeten dicha selección y se muestren de forma acorde.

6.3. GESTOS DE PUNTERO

Los dispositivos móviles permiten formas de interacción avanzadas que no están disponibles en entornos de escritorio no táctiles. Desde el uso de gestos multipunto en una pantalla táctil hasta la interacción mediante el movimiento u orientación del dispositivo.

En la UNE-EN 301549:2022 no se impide emplear todas estas formas de interacción en un sitio o aplicación web móvil. Sin embargo, han de realizarse de forma que no sean un obstáculo para las personas con discapacidad.

Así, toda la funcionalidad que emplee gestos multipunto o dependientes del trazo realizado también se debe poder realizar empleando un único punto de contacto y sin trazos, mediante una pulsación sencilla con un dedo o con un puntero. Por ejemplo, si es posible hacer un gesto de pinza con dos dedos para hacer zoom sobre un mapa o un movimiento lineal para desplazarse sobre el mismo entonces también debe haber un par de botones que permitan hacer zoom y otros botones (p. ej. rueda de cuatro puntos) que permitan el desplazamiento. El objetivo de este criterio es asegurar que el contenido se puede operar desde un gran número de dispositivos de entrada sencillos de forma que las personas con problemas de movilidad los puedan realizar.

En relación con este punto, es importante recordar que también se aplica a los dispositivos móviles lo indicado en el apartado 5.7.3 Cancelación del puntero. Es decir, las acciones se deben lanzar cuando se levanta el dedo de la pantalla en lugar de directamente cuando se pulsa. De esta forma se pueden evitar acciones indeseadas cuando se toca la pantalla accidentalmente.

6.4. ACTUACIÓN POR MOVIMIENTO

Cualquier funcionalidad que se pueda operar mediante el movimiento del dispositivo también se debe poder realizar a través del interfaz de usuario y también se debe poder desactivar dicha operación mediante el movimiento para evitar acciones no deseadas. Esto beneficia a las personas que tienen sus dispositivos anclados en posiciones fijas (p. ej. una silla de ruedas) y no pueden interactuar mediante el movimiento. También se evita que personas con problemas de movilidad ejecuten ciertas acciones accidentalmente al realizar movimientos no controlados.

7. FORMULARIOS EN HTML5

HTML5 incorpora importantes novedades en los formularios enfocadas tanto a mejorar la experiencia de los usuarios, enriquecer el conjunto de componentes de interacción existente y simplificar el desarrollo al incorporar de forma nativa características que antes necesitaban ser programadas.

7.1. NUEVOS TIPOS DE CAMPOS DE FORMULARIO

A la lista de los campos de formulario ya existentes, en HTML5 se les añade un número importante de nuevos campos de formulario, en especial nuevos tipos para el elemento `INPUT`.

El catálogo de tipos de `INPUT` se ve considerablemente aumentado en HTML5 con 13 nuevos campos para introducir tipos de datos manejados habitualmente en los formularios y aplicaciones web. Así, se definen campos específicos para introducir direcciones de correo electrónico, direcciones URLs, términos para búsquedas, números de teléfono, datos numéricos, fechas y colores.

Dependiendo del tipo de campo, las ventajas que aportan pueden ser de dos tipos:

- Soporte de las nuevas características de los formularios en HTML5 como puede ser la validación automática de forma nativa en los navegadores. Por ejemplo, los navegadores con soporte de HTML5 validarán que las direcciones de correo electrónico y URLs estén bien formadas o que los datos numéricos cumplan las restricciones indicadas.

- Componentes de interacción (widget) con una presentación específica para introducir el dato correspondiente. Por ejemplo, los navegadores que lo soporten permitirán introducir las fechas mediante un calendario o, colores mediante selectores de color o números mediante widgets de tipo *spinbox* o *slider*.

Figura 15. Widgets de tipo spinbox y slider para campos de tipo number y range (en Chrome)

Existen 13 nuevos tipos de `INPUT` para formularios, los cuales se listan en la siguiente tabla donde se indica además el tipo de datos que representan y el tipo de control que mostrarán los navegadores a los usuarios.

Tabla 3. Nuevos tipos de controles INPUT para formularios

VALOR	TIPO DE DATO	TIPO DE CONTROL
email	Una dirección de correo electrónico o una lista de correos electrónicos	Campo de texto
url	Una dirección URL	Campo de texto
search	Texto para una búsqueda (texto sin saltos de línea)	Campo de texto
tel	Un número de teléfono (texto sin saltos de línea)	Campo de texto
number	Un valor numérico	Campo de texto o control de tipo "spinner"
range	Un valor numérico dentro de un determinado rango, con el significado semántico añadido de que no es importante un valor exacto	Control tipo "slider" o similar
date	Una fecha (año, mes, día) sin zona horaria	Control de fecha
month	Una fecha formada por un año y un mes, sin zona horaria	Control de fecha (mes)

VALOR	TIPO DE DATO	TIPO DE CONTROL
week	Una fecha formada por un número de semana del año y un número de semana, sin zona horaria	Control de fecha (semana)
time	Una hora (hora, minuto, segundo, fracción de segundo) sin zona horaria	Control de hora
datetime	Una fecha y hora (año, mes, día, hora, minuto, segundo, fracción de segundo) con la zona horaria establecida en UTC (Tiempo Universal Coordinado)	Control de fecha y hora
datetime-local	Una fecha y hora (año, mes, día, hora, minuto, segundo, fracción de segundo) sin zona horaria	Control de fecha y hora
color	Un color sRGB compuesto por rojo, verde y azul en 8 bits. Color RGB en valor hexadecimal de seis dígitos (p. ej. #CCFF00)	Selector de color

Además de los nuevos tipos de `INPUT` también se definen otros nuevos elementos de formulario que enriquecen el catálogo de componentes.

Por ejemplo, el elemento `PROGRESS` para indicar visualmente el progreso de una tarea que requiere cierto tiempo (barra de progreso); el elemento `METER` para indicar visualmente el valor de una medida (gauge o gráfico de barra); o el elemento `DATALIST` que, asociado a elementos `INPUT`, permite su autocompletado con valores predeterminados.

Figura 16. Barra de progreso para el elemento PROGRESS (en Chrome)

Algunos de estos nuevos elementos no están relacionados con la entrada de datos sino con la salida de información hacia los usuarios, mejorando la usabilidad al mismo tiempo que se mejora su marcado semántico.

Como vemos, con HTML5 se enriquece la interfaz de los formularios con nuevos componentes de interacción que antes sólo se podían crear mediante programación *ad hoc* o el uso de frameworks de JavaScript. Se mejora la usabilidad y se facilita el desarrollo al poder usar de forma nativa dichos componentes.

7.2. VALIDACIÓN AUTOMÁTICA Y OTRAS NUEVAS CARACTERÍSTICAS

Como ya se ha comentado, una de las principales novedades de HTML5 es la posibilidad de validar los campos de formulario de forma automática antes de enviarlos. Es una validación realizada de forma nativa en el navegador que puede sustituir a la validación en cliente mediante scripts.

Los navegadores que soporten la validación automática mostrarán mensajes de error si no se respeta el formato de los tipos de datos introducidos (por ejemplo, emails, URLs, números, etc.)

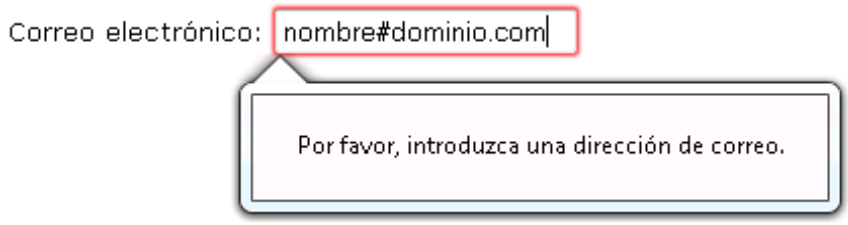

Figura 17. Mensaje de error al validar de forma automática un campo de tipo email (Firefox)

Esta validación automática no sólo abarca al formato de los datos introducidos. En HTML5 es posible identificar los campos obligatorios (atributo `required`) para que al enviar el formulario el navegador verifique si dichos campos se han rellenado y avise a los usuarios en caso contrario.

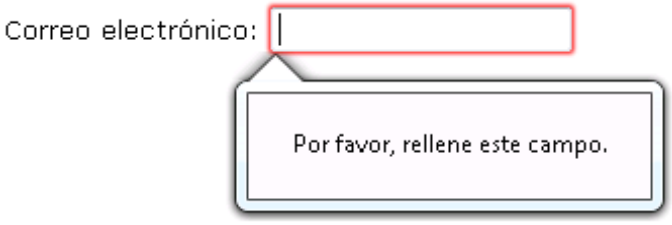

Figura 18. Aviso de campo obligatorio sin rellenar (Firefox)

Adicionalmente, es posible especificar restricciones personalizadas mediante expresiones regulares (atributo *pattern*) en caso de necesidad si nos es insuficiente con la validación de los tipos de datos por defecto, ampliando considerablemente las posibilidades de la validación automática.

La validación automática de HTML5 nunca podrá sustituir a la validación en servidor, pero sí podrá sustituir a la validación realizada en cliente mediante scripts.

No podemos confiar en el soporte de HTML5 mientras este no sea completo, igual que no podemos confiar en la ejecución de los scripts para la validación en cliente. Por tanto, si se quiere asegurar la integridad de los datos es necesario seguir realizando la validación en el servidor.

Otras características nuevas e interesantes de los formularios son la posibilidad de que un campo reciba el foco automáticamente (atributo *autofocus*) al cargar una página web o el incluir en determinados campos del formulario un texto de relleno (atributo *placeholder*) que desaparezca automáticamente al recibir el foco.

7.3. DISEÑAR PARA NAVEGADORES SIN SOPORTE DE HTML5

El soporte actual de los nuevos elementos de formulario de HTML5 es todavía limitado en algunos navegadores de forma que no se pueden aprovechar sus ventajas en los navegadores sin soporte.

A pesar de este soporte aún limitado, es posible usar los nuevos elementos de formulario de HTML5 de forma compatible con los navegadores sin soporte.

Los navegadores que no soporten los nuevos tipos de `INPUT` los interpretarán y mostrarán como elementos de tipo texto (`INPUT` de tipo *text*). Se perderán las nuevas características como la validación automática, pero siempre se podrá realizar también mediante JavaScript o mediante validación en el servidor, al igual que si se tratasen de formularios HTML4.

Otros elementos, como `PROGRESS` o `METER`, permiten proporcionar un contenido alternativo en forma de texto para que lo muestren los navegadores que no los soporten.

Los navegadores sin soporte se comportarán como hasta ahora con los formularios HTML4, mientras que los navegadores modernos se aprovecharán de las ventajas que aportan las nuevas características de los formularios de HTML5.

Sin embargo, también es posible usar los nuevos elementos de HTML5 para que los navegadores con soporte saquen ventaja de forma nativa de sus características mientras proporcionamos una alternativa equivalente en funcionalidad para los navegadores sin soporte, de forma que la apariencia final sea similar en todos los navegadores.

Para ello es necesario emplear JavaScript mediante una variante de la técnica de *mejora progresiva*. Básicamente se trata de que cuando se quiera emplear alguna característica de HTML5 realizar los siguientes pasos:

- Incluir el marcado de HTML5 que se desea emplear.
- Usar JavaScript para detectar el soporte de la característica HTML5 empleada.
- Si el navegador no soporta la característica de HTML5, proporcionar una alternativa equivalente en funcionalidad para los navegadores que no la soporten.

Aunque existen varias técnicas que se pueden usar para la detección de características de HTML5, existen librerías de JavaScript creadas con esa finalidad y que se puede hacer uso de ellas.

Una de estas librerías es Modernizr. Esta librería detecta el soporte de las características de HTML5 y CSS3. Para usarla basta con incluir en la cabecera del documento (`HEAD`) el archivo JavaScript correspondiente.

A esta técnica también se la conoce como *mejora regresiva* en lugar de mejora progresiva porque se emplean por defecto las nuevas características de HTML5 y se proporciona una alternativa equivalente en HTML4+JavaScript para los navegadores que no las soporten. Con el paso del tiempo y el aumento del soporte de HTML5, menos navegadores dependerán de la alternativa y más navegadores usarán el código HTML5 usado por defecto.

El uso de JavaScript no supone un problema de accesibilidad en el contexto de la norma UNE-EN 301549 siempre que el JavaScript se use de forma accesible.

8. HERRAMIENTAS DE EVALUACIÓN DE ACCESIBILIDAD

A la hora de construir el sitio web, crear plantillas y generar nuevos contenidos y servicios en las Sedes electrónicas, es importante realizar comprobaciones de accesibilidad para verificar que no se introducen problemas que puedan suponer una barrera de acceso para las personas con discapacidad.

Para ello existen diversas herramientas que se pueden usar como ayuda a la hora de realizar evaluaciones de forma manual, así como herramientas que realizan validaciones de accesibilidad de forma automática.

Sin embargo, como para evaluar la accesibilidad de los contenidos de una Sede electrónica normalmente se requiere autenticación algunas de las herramientas no se pueden emplear porque se tratan de servicios online. Estas herramientas online suelen ser precisamente los validadores automáticos.

Por tanto, en estos entornos estamos limitados a aquellas herramientas que se puedan usar de forma local, en intranets o en áreas de acceso restringido. Estas herramientas suelen ser aplicaciones de escritorio o bien extensiones de diversos navegadores que al ejecutarse sobre lo que muestra el navegador no tienen restricciones de acceso. Sin embargo, la mayoría de estas herramientas no realizan comprobaciones automáticas, aunque sí son de ayuda para realizar las evaluaciones manuales.

En la ***"Guía de validación de accesibilidad web"***, disponible en el área de documentación del Portal de la Administración Electrónica (PAe), se puede consultar un amplio listado de herramientas que se pueden usar para evaluar la accesibilidad de un sitio web, tanto herramientas automáticas como manuales, indicando para cada una de ellas si se pueden usar de forma local o únicamente funcionan online.

9. GUÍA RÁPIDA PARA DESARROLLADORES

A continuación, se proporciona como resumen una serie de recomendaciones que deben tenerse en cuenta cuando se implementen servicios de tramitación telemática en Sedes electrónicas:

- La utilización de firma digital o/y certificados electrónicos no impide que un trámite sea accesible siempre que la tecnología empleada (JavaScript, applet, ActiveX, etc.) se use de forma accesible y compatible con los productos de apoyo. Así, por ejemplo, la interfaz de la herramienta cliente (si la hubiera) deberá ser directamente accesible. En todo caso, también sería recomendable proporcionar otras vías de tramitación para aquellos usuarios que no puedan utilizar la firma electrónica.

- Los formularios deben poseer características de accesibilidad adecuadas, **asociando explícitamente** las etiquetas, cuando existan, con sus controles de formulario o al menos proporcionando un título descriptivo a los campos que carezcan de ellas. Asimismo, se ha de colocar y marcar adecuadamente la **información adicional** en el mismo, **agrupándose** correctamente, etc. Por otra parte, 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 (p. ej. para facilitar el autocompletado).

- En los componentes del interfaz de usuario, como los campos de formulario o botones, es necesario que el **texto visible** que actúa como su etiqueta y que sirve para reconocerlos visualmente también **forme parte de su nombre accesible**.

- Se debe informar a los usuarios acerca de todos los **errores de validación** producidos al introducir datos en un formulario. Dichos avisos se deberán mostrar de manera accesible y **en formato de texto** antes del formulario de forma que no pasen inadvertidos para el usuario. Cuando sea posible, se proporcionarán sugerencias de corrección para los errores producidos.

- Se debe proporcionar la **información y ayuda** que sea necesaria para que los usuarios puedan rellenar los formularios correctamente sin cometer errores.

- Cuando una transacción tenga implicaciones legales, económicas o de pérdida de información personal de los usuarios, dar al menos la posibilidad de **revisar la información antes de enviarla** para que los usuarios puedan **confirmar los datos o corregir** los posibles errores detectados.

- Todos los elementos de interacción, incluidos los formularios, deben ser **independientes de dispositivo** y poder ser seleccionados y activados a través del teclado o mediante productos de apoyo, con un orden de tabulación correcto e indicando visualmente la posición del foco del teclado.

- Para facilitar la interacción a las personas que emplean magnificadores de pantalla, cuando se muestre un contenido adicional como reacción al *hover* del ratón o al foco del teclado (p. ej. un menú desplegable que tape parte de la página) entonces el usuario debe poder ocultar el contenido si así lo desea sin necesidad de mover el ratón o el foco, o bien ha de poder interactuar adecuadamente con el contenido desplegado.

- No provocar cambios de contexto (nueva pestaña, ventana, etc.) cuando se cambie el estado o valor de un control de formulario, o cuando un control reciba el foco del teclado, a no ser que previamente se avise a los usuarios sobre este comportamiento.

- Identificar como regiones activas de WAI-ARIA los **mensajes informativos o de estado** (p. ej. mensajes de error, notificaciones o avisos) que se muestren dinámicamente en una ubicación de la página diferente de la actual del usuario para que puedan ser reconocidos por los lectores de pantalla y transmitidos automáticamente al usuario.

- El contenido de los sitios web debe poder presentarse sin pérdida de información o funcionalidad y sin necesidad de realizar scroll en dos dimensiones en pantallas pequeñas. Es decir, la página se **debe "adaptar" correctamente a diferentes tamaños de ventana** mediante una maquetación fluida, un diseño *responsive* o similar.

- El sitio web se debe poder **usar sin problemas desde dispositivos móviles por personas con discapacidad** sin restringir su visualización y funcionalidad a una única orientación de la pantalla (horizontal o vertical) y sin necesidad de interactuar mediante gestos multipunto (varios dedos),

gestos dependientes del trazo o mediante el movimiento del dispositivo (orientación).

- Se debe garantizar el cumplimiento de todos los requisitos de accesibilidad aplicables a cualquier sitio web. Algunos de ellos presentes con mayor frecuencia en las sedes serían:
  - Identificar los encabezados en tablas de datos, así como las relaciones entre celdas de datos y encabezados.
  - Identificar y marcar correctamente las listas.
  - Identificar y marcar correctamente los títulos o encabezados de página.
  - Proporcionar textos alternativos a las imágenes empleadas.
  - Hay que asegurar que haya un contraste suficiente entre el texto y el fondo, entre el texto mostrado en imágenes y el fondo, así como en aquellos contenidos gráficos cuya información visual es necesaria para su identificación o comprensión como pueden ser los botones, iconos, diagramas, gráficas, infografías, etc.
  - Identificar correctamente el idioma principal de la página y los cambios de idioma
  - Dotar a los enlaces de un texto suficientemente descriptivo de su destino o funcionalidad.
  - Se recomienda emplear unidades relativas en el diseño de las páginas.
  - Se recomienda utilizar hojas de estilo CSS externas para aplicar efectos de presentación.
  - Respecto a la presentación del texto, los estilos han de realizarse de forma que no se pierda contenido o funcionalidad cuando los usuarios modifiquen algunas de las características del texto como el espaciado entre líneas, párrafos, letras y palabras.
  - Emplear un código (X)HTML y CSS válido gramaticalmente o, al menos, que esté *"bien formado"*.

- Generar documentos PDF accesibles.
- Utilizar tiempos de sesión apropiados y no limitar el tiempo de lectura o interacción (movimientos de contenido, actualizaciones o refrescos automáticos, etc.)