DESARROLLO DE APLICACIONES MULTIPLATAFORMA

DESARROLLO DE APLICACIONES WEB

ADMINISTRACIÓN DE SISTEMAS INFORMÁTICOS EN RED

Lenguajes de marcas y sistemas de gestión de información

Fernando Paniagua Martín

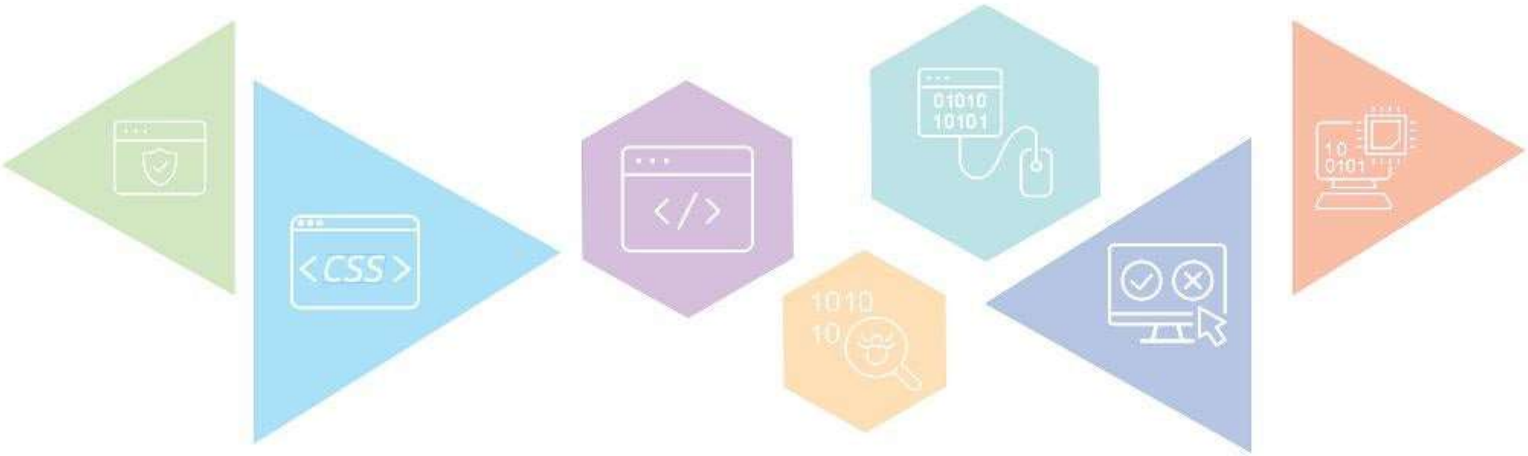
Incluye recursos digitales en www.paraninfo.es

Paraninfo
ciclos formativos

Lenguajes de marcas y sistemas de gestión de información

Lenguajes de marcas y sistemas de gestión de información

Fernando Paniagua Martín



Paraninfo
ciclos formativos

Paraninfo

Lenguajes de marcas y sistemas de gestión de información

© Fernando Paniagua Martín

Gerente Editorial
María José López Raso

Técnico Editorial
Paola Paz Otero
Sofía Durán Tamayo

Editora de Adquisiciones
Carmen Lara Carmona

Producción
Nacho Cabal Ramos

Diseño de cubierta
Ediciones Nobel

Preimpresión
Montytexto

Reservados los derechos para todos los países de lengua española. De conformidad con lo dispuesto en el artículo 270 del Código Penal vigente, podrán ser castigados con penas de multa y privación de libertad quienes reprodujeren o plagiaren, en todo o en parte, una obra literaria, artística o científica fijada en cualquier tipo de soporte sin la preceptiva autorización. Ninguna parte de esta publicación, incluido el diseño de la cubierta, puede ser reproducida, almacenada o transmitida de ninguna forma, ni por ningún medio, sea este electrónico, químico, mecánico, electroóptico, grabación, fotocopia o cualquier otro, sin la previa autorización escrita por parte de la Editorial.

Todas las marcas comerciales y sus logos mencionados en este texto son propiedad de sus respectivos dueños.

COPYRIGHT © 2021 Ediciones Paraninfo, SA
1.ª edición, 2021

C/ Velázquez 31, 3.º Dcha. / 28001 Madrid ESPAÑA
Teléfono: 914 463 350 / Fax: 914 456 218
clientes@paraninfo.es / www.paraninfo.es

ISBN: 978-84-283-4059-5
Depósito legal: M-12522-2021

(16.596)

Impreso en España /Printed in Spain
Gráficas Eujoa, (Meres, Asturias)

La editorial recomienda que el alumnado realice las actividades sobre el cuaderno y no sobre el libro.

Presentación

Este libro desarrolla los contenidos del módulo profesional de **Lenguajes de marcas y sistemas de gestión de información** de los Ciclos Formativos de grado superior de Desarrollo de Aplicaciones Multiplataforma y Desarrollo de Aplicaciones Web, pertenecientes a la familia profesional de Informática y Comunicaciones.

Las unidades del libro se acompañan de multitud de **recursos didácticos** que ayudarán al futuro profesional a comprender la materia y acercarlo a su inminente realidad laboral:

- Mapas conceptuales, actividades tipo test, de aplicación y de ampliación al final de cada unidad.
- Gran cantidad de actividades propuestas y resueltas intercaladas con la teoría.
- Argot técnico.

Este libro dispone de los siguientes **materiales y recursos** disponibles en línea para el **profesorado que confirme su adopción** como libro de texto para impartir la materia:

- **Programación didáctica.**
- **Solucionario.**
- **Presentación en PowerPoint.**
- **Examina.**
- **LDP (Libro Digital Proyectable).**

Este libro dispone además de **material previo registro** disponible en línea para el **alumnado:**

- **Videotutoriales para guiar en la ejecución de actividades prácticas contenidas en el temario.**

© Ediciones Paraninfo

Índice

1. Introducción a los lenguajes de marcas 1

Introducción. 2
1.1. Introducción a los lenguajes de marcas 2
1.2. Contexto de los lenguajes de marcas 4
1.3. Lenguajes de marcas: características, estructura y formatos 7
1.4. Lenguajes de marcas y dialectos más comunes 9
1.4.1. HTML 9
1.4.2. XML 10
1.4.3. XHTML. 11
1.4.4. JSON. 13
1.4.5. SVG. 14
1.4.6. MathML. 14
1.4.7. NewsML 14
1.4.8. FXML. 14
1.4.9. XAML. 15
1.4.10. Android 15
1.4.11. Servicios web. 16
1.4.12. RSS y Atom 16
1.4.13. XSL. 17
Mapa conceptual 18
Actividades finales 19

2. Los lenguajes de la web. HTML y XHTML. 23

Introducción. 24
2.1. Introducción, evolución y estado actual de HTML 24
2.2. Estructura y sintaxis del lenguaje HTML. 27
2.2.1. Estructura y sintaxis 27
2.2.2. Entidades 30
2.2.3. Comentarios 31
2.3. Elementos de HTML 32
2.3.1. Elemento html 34
2.3.2. Elemento head. 35
2.3.3. Elemento body 38
2.3.4. Elementos de bloque . . . 38
2.3.5. Elementos de línea. 48
2.3.6. Contenido incrustado . . . 63

© Ediciones Paraninfo

2.4. Tablas en HTML: estructura y elementos 72
2.4.1. Elementos de las tablas 72
2.5. Formularios en HTML: estructura y elementos 79
2.5.1. Elementos de los formularios 79
2.5.2. Validaciones 94
2.6. Multimedia en HTML 97
2.6.1. Elementos multimedia . . 98
2.7. XHTML: la versión XML de HTML 105
2.7.1. Fundamentos 106
2.7.2. Diferencias entre XHTML y HTML 106
2.7.3. Representación de XHTML 107
2.8. Herramientas de consulta y edición de HTML 109
2.8.1. Navegadores web 109
2.8.2. Editores de texto 111
2.8.3. Editores visuales de HTML 112
2.8.4. Editores online 112
Mapa conceptual 114
Actividades finales 115

3. Definición de estilos en entornos web: CSS 121

Introducción 122
3.1. Introducción, evolución y estado actual de CSS 122
3.2. Estructura y sintaxis de CSS . . . 123
3.3. Aplicación de CSS 124
3.4. Prioridades y orden en CSS 127
3.5. Valores y unidades en CSS 129
3.5.1. Tipos de datos 129
3.5.2. Unidades de longitud absoluta 130
3.5.3. Unidades de longitud relativas 130
3.5.4. Colores 132
3.5.5. Imágenes 133
3.5.6. Tipos de letras 135
3.6. Selectores CSS 137
3.6.1. Selectores básicos 138
3.6.2. Agrupación de selectores 140
3.6.3. Combinadores 141
3.6.4. Pseudoclases 142
3.6.5. Pseudoelementos 147
3.7. Propiedades CSS 149
3.7.1. Contenedores y posicionamiento: el modelo de cajas 149
3.7.2. Propiedad float 156
3.7.3. Propiedad position 159
3.7.4. Propiedad overflow y el contenido desbordante 162
3.7.5. Propiedades de texto . . 163
3.7.6. Propiedades de listas . . 165
3.7.7. Propiedades de tablas . . . 165
3.7.8. CSS aplicado a formularios 166
3.7.9. CSS aplicado a enlaces 168
3.7.10. Fondos, bordes y contornos 168
3.7.11. Diseño adaptativo con *media queries* 169
Mapa conceptual 171
Actividades finales 172

4. XML: creación, validación y utilización 177

Introducción 178
4.1. Introducción, evolución y estado actual 178
4.2. Estructura y sintaxis de XML . . . 179
4.2.1. Declaración 180
4.2.2. Estructura, sintaxis, elementos y atributos . . . 180
4.2.3. Comentarios 185
4.2.4. Espacios de nombres . . . 185
4.2.5. Entidades 186
4.2.6. CDATA 188

© Ediciones Paraninfo

4.3. Validación de XML 189
4.3.1. XML válido y XML bien formado 189
4.3.2. DTD. 190
4.3.3. XML Schema 203
4.3.4. Otras formas de validación 217
4.4. XML aplicado 217
4.4.1. XML en aplicaciones. . . . 217
4.4.2. XML en la web 218
4.4.3. XML en servicios web . . . 219
4.5. Herramientas de visualización y edición de XML 220
4.5.1. Editores de texto 220
4.5.2. Navegadores web 221
4.5.3. XML Notepad 222
4.5.4. Liquid Studio 224
4.5.5. XMLPad 224
4.5.6. Eclipse 226
4.5.7. Editores web 227
Mapa conceptual 228
Actividades finales 229

5. Almacenamiento de información 235

Introducción. 236
5.1. Creación de documentos XML desde programa 236
5.2. Sistemas de almacenamiento de documentos XML 240
5.2.1. Recuperación de documentos XML desde fichero. 241
5.2.2. Almacenamiento de documentos XML en fichero 248
5.2.3. Almacenamiento de documentos XML en base de datos convencional 249
5.2.4. Almacenamiento de documentos XML en base de datos nativa XML. 251
5.3. Técnicas de búsqueda de información en documentos XML: lenguajes de consulta y manipulación 257
5.3.1. XPath 258
5.3.2. XQuery 269
5.3.3. XLink y XPointer 274
Mapa conceptual 275
Actividades finales 276

6. Conversión y adaptación de documentos XML. 281

Introducción. 282
6.1. La familia de lenguajes de hojas de estilo extensibles: XSL 282
6.2. XSLT 283
6.2.1. Estructura 283
6.2.2. Transformación 284
6.2.3. Elementos de XSLT. 288
6.3. XSL-FO. 294
6.3.1. Fundamentos. 294
6.3.2. Estructura y sintaxis 298
6.4. Herramientas de transformación XML. 303
6.4.1. Navegadores web 303
6.4.2. XMLPad 304
6.4.3. XML Notepad 305
6.4.4. Editores en línea 305
6.4.5. Procesadores a través de programa 306
Mapa conceptual 307
Actividades finales 308

7. JSON, el lenguaje ligero. Creación y utilización 313

Introducción. 314
7.1. Introducción a JSON 314
7.2. Estructura y sintaxis del formato JSON 316
7.3. Aplicaciones de JSON 321
7.3.1. Configuración de aplicaciones 321

© Ediciones Paraninfo

7.3.2. Intercambio de datos y servicios web. 322
7.3.3. Almacenamiento 324
7.4. Herramientas de edición y consulta de documentos JSON 326
7.4.1. Editores de escritorio . . . 326
7.4.2. Editores en línea 326
7.4.3. Navegadores 327
Mapa conceptual 328
Actividades finales 329

8. Los lenguajes de marcas como herramienta de redifusión de información. . . 333

Introducción. 334
8.1. Introducción a los sistemas de redifusión de información . . . 334
8.2. Formato RSS 335
8.3. Formato Atom. 338
8.4. Editores de canales de redifusión 342
8.5. Agregadores de contenidos 342
8.6. Navegadores web y otras herramientas 344
8.7. Canales RSS y Atom 346
8.7.1. Prensa general. 346
8.7.2. Prensa deportiva 347
8.7.3. Inversión 347
8.7.4. Organismos oficiales y entes públicos. 347
8.7.5. AEMET. 347
Mapa conceptual 348
Actividades finales 349

9. Sistemas de gestión de información 353

Introducción. 354
9.1. Introducción a los sistemas de gestión de información 354
9.2. ERP 355
9.2.1. Características de los ERP. 356
9.2.2. Tipos de ERP 357
9.2.3. Beneficios aportados por los ERP 358
9.2.4. Fabricantes de ERP más populares. 358
9.2.5. ERP gratuitos. 359
9.3. CRM 360
9.3.1. Características de los CRM 361
9.3.2. Beneficios aportados por los CRM. 361
9.3.3. CRM más populares. . . . 362
9.4. BI 363
9.4.1. ETL 364
9.4.2. *Data Warehouse* 365
9.4.3. OLAP. 365
9.4.4. *Data Mining* 366
9.4.5. Cuadro de mandos. 367
9.4.6. Informes 369
9.4.7. Sistemas de BI más populares. 369
9.5. Otros sistemas de gestión de información 370
9.6. *SaaS* y *Cloud Computing* 370
Mapa conceptual 372
Actividades finales 373

© Ediciones Paraninfo

UNIDAD 1

Introducción a los lenguajes de marcas

Objetivos

- Descubrir el significado del término «lenguaje de marcas».
- Conocer el origen y la motivación que llevó a crear los lenguajes de marcas actuales.
- Concretar los lenguajes de marcas más utilizados en la actualidad y cuál es el uso principal de cada uno de ellos.
- Exponer la estructura básica de los lenguajes de marcas más comunes.
- Enumerar algunas de las herramientas disponibles para consultar o editar ficheros escritos con los lenguajes de marcas utilizados habitualmente.

Contenidos

- 1.1. Introducción a los lenguajes de marcas
- 1.2. Contexto de los lenguajes de marcas
- 1.3. Lenguajes de marcas: características, estructura y formatos
- 1.4. Lenguajes de marcas y dialectos más comunes

Introducción

Los lenguajes de marcas existen prácticamente desde el comienzo de la informática. Cubren la necesidad de disponer de lenguajes sencillos para el intercambio de información entre distintos sistemas. Con la llegada de internet, algunos de los lenguajes de marcas se convirtieron en piedras angulares sobre las que construir la *World Wide Web*.

Existe un gran número de lenguajes de marcas distintos y, obviamente, no todos son igual de importantes. No obstante, comparten algunos aspectos comunes: fundamentos, orígenes, características básicas y elementos relacionados con su estructura. Es importante conocer qué son, cómo han evolucionado y, de aquellos que han tenido más éxito, qué pueden proporcionar.

1.1. Introducción a los lenguajes de marcas

Un **lenguaje de marcas** o **lenguaje de marcado** es aquel que permite representar información que contiene, además de los datos propiamente dichos, marcas o etiquetas que indican cómo se estructuran estos datos, su significado o cómo debe representarse desde un punto de vista gráfico o visual.

Los lenguajes de marcas no son lenguajes de programación. No contienen instrucciones ni implican acciones directas. Son reglas que organizan la información con el objetivo de darle una estructura uniforme y facilitar su procesamiento automático.

Una clasificación simplificada de los lenguajes de marcas podría ser la siguiente:

- Lenguajes orientados a la presentación de información.
- Lenguajes orientados al almacenamiento y procesado de información.

El ejemplo más representativo de un lenguaje de marcas orientado a la presentación de información es el lenguaje de marcas de hipertexto *(HyperText Markup Language),* conocido como HTML.

El etiquetado de HTML indica al programa que va a presentar en la pantalla la información (generalmente un navegador web o *Web Browser*) cómo debe hacer dicha presentación. Es decir, proporciona junto a los datos que va a mostrar información referente a cómo han de mostrarse. Aunque ya existían previamente, a partir de la versión 5 de HTML se incluye un importante número de etiquetas con lo que se denomina **información semántica.** Estas etiquetas indican, además de cómo ha de mostrarse la información, qué significado tiene desde un punto de vista estructural. En la Unidad 2 se profundizará en el conocimiento de este tipo de etiquetado y las diferentes alternativas que proporciona.

Para ilustrar el lenguaje HTML, en el siguiente código se puede observar un texto delimitado por la etiqueta de apertura *<h1>* y la etiqueta de cierre correspondiente *</h1>*:

```
<h1>El origen de la vida</h1>
```


© Ediciones Paraninfo

Estas etiquetas o marcas indican que el texto «marcado» es una cabecera del más alto nivel. A nivel visual, la etiqueta *<h1>* indica al navegador web que el texto debe representarse de una forma determinada. Los diseñadores de los navegadores, a partir del estándar publicado por el W3C, saben que la representación debe ser como se muestra en la Figura 1.1.

El origen de la vida

Figura 1.1. *Representación de un elemento HTML etiquetado con <h1>.*

Por otra parte, la etiqueta *<h1>* indica que, semánticamente, el texto contenido es el título de un bloque de información del más alto nivel posible, como pueda ser el titular de una noticia en un periódico o el título de un capítulo de un libro.

HTML no es sino uno más de los múltiples lenguajes de marcas existentes. No existe un lenguaje de marcas universal, sino que pueden definirse tantos lenguajes de marcas como se desee, aunque probablemente ya exista algún lenguaje que proporcione la funcionalidad suficiente para casi cualquier necesidad.

Por ejemplo, un lenguaje de marcas sencillo podría definirse de la siguiente manera:

- Los elementos delimitados por la marca <#> son títulos de películas y deberán representarse en color azul y utilizando la tipografía *Times New Roman* con un tamaño de 16 puntos.
- Los elementos delimitados por el símbolo <%> son nombres de director y deberán representarse en color rojo y utilizando la tipografía *Times New Roman* con un tamaño de 14 puntos.

Dado este formato, un documento creado con este nuevo lenguaje de marcas podría ser el siguiente:

```
<#>Tron<#>
<%>Steven Lisberger<%>
<#>Juegos de guerra<#>
<%>John Badham<%>
```

Gracias a las marcas o etiquetas, un programa informático realizado según esta estructura puede saber cuál es el título y el director de una película de manera sencilla. También se puede saber cómo debe representarse la información contenida en el documento, ya que así se indica en la especificación del lenguaje (Figura 1.2).

Tron
Steven Lisberger
Juegos de guerra
John Badham

Figura 1.2. *Simulación de representación del etiquetado definido.*

© Ediciones Paraninfo

Se puede observar, por lo tanto, que la definición de un lenguaje de marcas está al alcance de cualquiera y no tiene por qué suponer mucho esfuerzo. El esfuerzo sería necesario en crear los programas que leyesen e interpretasen las etiquetas correctamente: habría que hacer tantos programas como lenguajes de marcas existentes.

Por fortuna, como se ha indicado con anterioridad, ya existe una serie de lenguajes de marcas capaces de cubrir prácticamente todas las necesidades posibles gracias a su flexibilidad. Además, no requieren la elaboración de sofisticados programas ni aplicaciones de procesamiento porque ya están creadas. Lenguajes como HTML y XML, junto con sus dialectos, están estandarizados y son capaces de satisfacer cualquier necesidad referente a cómo se deben estructurar los datos o cómo deben representarse.

1.2. Contexto de los lenguajes de marcas

El origen de los lenguajes de marcas se remonta prácticamente al origen de la computación moderna. En el mismo momento en que se comenzó a querer almacenar información para procesarla por un programa informático, se detectó la necesidad de dar «formato» a dicha información. Un dato etiquetado con una determinada marca facilita su proceso automático.

El problema surge cuando cada programa utiliza su propio lenguaje, ya que obliga a crear un conversor entre cada par de programas con los que se desea compartir información.

Figura 1.3. *Flujo de conversión de datos entre programas que utilizan distintos lenguajes.*

Resulta evidente que esta problemática impide compartir información de forma ágil y sencilla, algo que en el mundo académico y empresarial suele ser un objetivo fundamental desde que se comenzaron a utilizar los ordenadores y los sistemas informáticos.

A finales de los años sesenta y principios de los setenta del siglo XX, el equipo liderado por Charles Goldfarb, investigador en la compañía IBM, desarrolló el que se considera el precursor de los lenguajes de marcas modernos: GML *(Generalized Markup Language)*. GML es el primer paso para conseguir unificar los formatos de representación de información, aunque no fuese diseñado para ello. No obstante, GML sienta las bases para definir los estándares de representación de información posteriores.

GML se diseñó para utilizarse en procesadores de texto, de tal manera que, en el documento escrito con este lenguaje, se indica mediante unas etiquetas o marcas cómo está estructurado el contenido y así indicar que un texto es un encabezado y cuál es su nivel de importancia, una lista y sus elementos o una tabla y su contenido en filas, columnas y celdas, por citar algunos ejemplos. Las marcas en GML están precedidas de dos puntos

© Ediciones Paraninfo

(:) y seguidas de un punto (.) y tienen el aspecto que se muestra en el siguiente código de ejemplo:

```
:h1.Capítulo 1:  Introducción a los lenguajes de marcas
:p.Características y origen de los lenguajes de marcas.
:ol.
:li.Introducción
:li.Contexto
:li.Tipos de lenguajes
:eol.
```

Sin pretender entrar en detalles de un formato de documento ya obsoleto y en desuso, merece la pena indicar que la marca *:h1.* indica que el texto que aparece a continuación corresponde a un encabezado del máximo nivel (nivel 1); la marca *:p.* indica, por su parte, que el texto al que hace referencia es un párrafo; las marcas *:ol.*, *:li.* y *:eol.* indican el comienzo de una lista no numerada, los elementos contenidos y el final de dicha lista respectivamente.

La aportación histórica de GML es evidente: no es necesario definir un nuevo formato en cada documento, ya que se puede utilizar uno ya existente si este es lo suficientemente flexible como para permitir representar cualquier tipo de información. A partir de esta definición del formato se puede construir todo un universo de aplicaciones para crear, editar y representar documentos y que estas puedan utilizarse en cualquier contexto.

Como se ha indicado, GML se creó de la mano de IBM, una de las empresas más importantes en la historia de la computación, por lo que el paso siguiente no podía ser otro que estandarizar el lenguaje para que no dependiese de un ente privado con las implicaciones de todo tipo que esto puede tener (económicas, legales y técnicas). Este hecho dio lugar a la creación de lenguaje SGML.

SGML *(Standard Generalized Markup Language)* es el estándar creado en 1986 por la ISO (International Organization for Standarization). Esta organización internacional y no gubernamental está constituida por varias organizaciones de estandarización nacionales y tiene como objetivo crear estándares de ámbito internacional dentro de diversos campos. El estándar que define el lenguaje SGML es el ISO 8879:1986.

En realidad, SGML no es un lenguaje como tal, ya que no especifica qué marcas deben utilizarse. SGML indica cómo deben ser las marcas de un lenguaje. Esto significa que SGML puede ser el origen de un lenguaje de marcas, lo que le convierte en lo que se denomina **metalenguaje.** HTML y XML, por ejemplo, son lenguajes de marcas creados a partir del estándar SGML, aunque hay que aclarar que la versión 5 de HTML tiene sus propias reglas y ya no está basado en este estándar.

XML se ha convertido en un lenguaje de intercambio de datos entre sistemas debido a su simplicidad, robustez y capacidad. Se puede afirmar que XML es la evolución natural de SGML, ya que también se puede considerar un metalenguaje, aunque el lenguaje de marcas más conocido sea HTML.

Tim Berners-Lee desarrolló la *World Wide Web* (conocida coloquialmente como web o por sus siglas www) entre 1989 y 1990 mientras trabajaba en la Organización Europea para

© Ediciones Paraninfo

la Investigación Nuclear (CERN). La web se presentó en 1991 y, aunque fue inicialmente concebida como una tecnología propia de la CERN, su ámbito de aplicación es universal. Con carácter público, la web vio la luz en 1993 y tiene una característica fundamental para consolidar un éxito incuestionable: es gratuita.

Figura 1.4. *Las siglas www corresponden a* World Wide Web, *también conocida como web.*

La web tiene como objetivo principal simplificar el acceso a la información, tanto desde un punto de vista formal (estructura) como funcional (acceso). Estas dos perspectivas consiguen que la información sea accesible desde cualquier sistema operativo, ya que utiliza un lenguaje de marcado estándar (HTML) y la obtención de información relacionada es sencilla (a través de los hipertextos).

El lenguaje de marcado de hipertexto o HTML es, por lo tanto, la piedra angular de la web. Permite que el intercambio de información entre distintos sistemas sea posible independientemente del software o del hardware que tengan dichos sistemas. HTML fue lanzado en 1992 y es el lenguaje de marcas más conocido y utilizado.

En la Figura 1.5 se puede observar la cronología y dependencias de los principales lenguajes de marcas.

Algunos de estos lenguajes ya están en desuso (GML) o su uso es referencial (SGML). Otros, en cambio, están en plena vigencia y continua actualización para cubrir las nuevas necesidades que surgen en su utilización.

Nota técnica

La versión 5 de HTML ya no está basada en SGML, pero sigue inspirándose en este metalenguaje para su definición.

© Ediciones Paraninfo

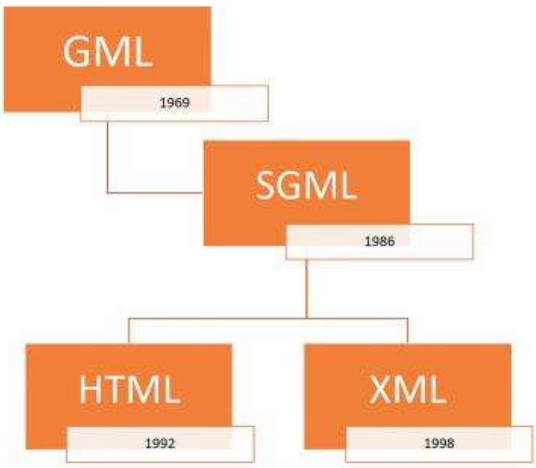


Figura 1.5. *Cronología de los lenguajes de marcas más usados.*

1.3. Lenguajes de marcas: características, estructura y formatos

Existen multitud de lenguajes de marcas y esto provoca que las estructuras y los formatos sean también numerosos. No existe, por lo tanto, una estructura común o un único formato. Por definición, cuando se habla de un lenguaje de marcas, se está haciendo referencia a la información añadida mediante marcas a los datos propiamente dichos del documento.

Cada lenguaje tiene su propia estructura y formato. No obstante, con el objetivo de ilustrar, se presenta a continuación parte de la estructura de uno de los lenguajes de marcas más importantes, que no es otro que XML. En la Unidad 4 se mostrará en detalle la estructura completa.

La estructura parcial y simplificada de un documento XML es la siguiente:

- Cada elemento de un documento XML está precedido por una etiqueta de apertura compuesta por un nombre y rodeada por los símbolos < y >.
- Cada elemento de un documento XML está seguido por una etiqueta de cierre, compuesta por el mismo nombre de la etiqueta de apertura y rodeada por los símbolos </ y >.
- El contenido de un elemento de un documento XML puede ser un texto, o uno o más elementos.
- Solo existe un elemento raíz.
- Un atributo está compuesto de un nombre del atributo, el símbolo = y el valor del atributo entrecomillado.
- Un elemento puede tener entre 0 y n atributos.

Por su parte, el formato en el que está escrito un documento XML es texto plano.

© Ediciones Paraninfo

Un ejemplo de un documento XML de acuerdo con esta estructura podría estar almacenado en un fichero denominado *misjuegos.xml* y tener el siguiente contenido:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<videojuegos>
    <videojuego multijugador="si">
        <titulo>Call Of Duty</titulo>
        <plataforma>Play Station 5</plataforma>
    </videojuego>
    <videojuego multijugador="si">
        <titulo>Fortnite</titulo>
        <plataforma>XBox One</plataforma>
    </videojuego>
    <videojuego multijugador="no">
        <titulo>Manic Miner</titulo>
        <plataforma>Amstrad CPC</plataforma>
    </videojuego>
</videojuegos>
```

Como se puede deducir, los nombres de las etiquetas y de las propiedades no están definidas en ningún estándar. Es el autor del documento quien decide cómo se han de llamar dichas etiquetas, cuál es su orden y cómo se estructuran entre sí. Esta flexibilidad es debida a que, como se ha comentado anteriormente, XML es un metalenguaje. En cualquier caso, la estructura ha sido respetada y eso convierte a este documento en un documento XML válido.

Este fichero, en resumen, está sujeto a una estructura y se almacena en un formato susceptible de ser leído y procesado por un programa escrito en cualquier lenguaje de programación y ejecutado en cualquier sistema operativo.

Por su parte, HTML, siendo también un lenguaje surgido a raíz de SGML, no es un metalenguaje como XML. Tiene también una estructura bien definida (similar a la de XML, pero no idéntica), aunque, al tratarse de un lenguaje concreto, tiene definidos con precisión los nombres de etiquetas y de los atributos.

En el siguiente ejemplo se muestra una sección de un documento HTML. Las etiquetas *<table>*, *<tr>* y *<td>* están definidas en la especificación del lenguaje HTML, así como el atributo *class*.

```
<table class="fondoGris">
    <tr>
        <td>Elemento 1</td>
        <td>Elemento 2</td>
    </tr>
    <tr>
        <td>Elemento 3</td>
        <td>Elemento 4</td>
    </tr>
</table>
```


© Ediciones Paraninfo

En las unidades 2 y 4 se profundizará en las especificaciones formales de estos dos lenguajes, así como de los dialectos derivados y de otros lenguajes relacionados.

Las características fundamentales de los lenguajes de marcas son las siguientes:

- **Independencia.** Su uso no debe depender de ninguna plataforma hardware ni sistema software concreto.
- **Almacenados en texto plano.** El documento está compuesto únicamente por caracteres de texto.
- **Flexibilidad.** Los lenguajes de marcas deben ser lo suficientemente flexibles para poder utilizarse en diferentes contextos.
- **Compactibilidad.** Las marcas y el contenido se encuentran en el mismo documento.
- **Facilidad de procesamiento.** El formato debe ser diseñado de tal manera que el procesamiento automático sea sencillo.

1.4. Lenguajes de marcas y dialectos más comunes

La web y las aplicaciones desarrolladas sobre la tecnología de internet han supuesto un importante incremento en el uso de los lenguajes de marcas tanto para representar información como para intercambiar datos entre sistemas. Además, existen diversos campos en los que los lenguajes de marcas tienen una utilidad importante, como pueden ser la configuración de aplicaciones o la definición de interfaces gráficas de usuario.

Actualmente, los lenguajes de marcas más habituales tienen como objetivos principales la representación de información (HTML y XHTML) o el almacenamiento e intercambio de información (XML y sus dialectos, y el formato de intercambio JSON).

En este apartado se van a presentar algunos de los lenguajes de marcas y dialectos de uso más común en la actualidad, pero no son los únicos. En XML, en concreto, existen multitud de dialectos para prácticamente cualquier campo del conocimiento.

1.4.1. HTML

Es el lenguaje de marcas más conocido y utilizado, y el W3C se encarga de su mantenimiento y estandarización. A la escritura de este libro, HTML se encuentra en la versión 5.2, con fecha 14 de diciembre de 2017. La versión 5.3, con fecha 18 de octubre de 2018, se halla en borrador y sujeta revisión y discusión *(Working Draft)*. Independientemente de la versión concreta, coloquialmente se hace referencia a todas las versiones 5.x (versiones 5.1, 5.2, 5.3. y las que puedan aparecer) de HTML como **HTML5.**

La primera versión de HTML5 se publicó en octubre de 2014 y supuso un cambio radical y muy exitoso en el lenguaje.

Su objetivo principal sigue siendo el concebido por Tim Berners-Lee: proporcionar un mecanismo para diseñar páginas web que van a ser representadas en un navegador, ya sea este ejecutado en un ordenador, en una videoconsola o en un dispositivo móvil.

© Ediciones Paraninfo

Figura 1.6. *HTML5 es la versión actual de HTML y goza de enorme popularidad.*

El éxito de HTML5 es indudable. Los diseñadores gráficos cuentan con una capacidad ilimitada de elementos de maquetación gracias al uso conjunto con las hojas de estilo en cascada (CSS).

También los programadores de aplicaciones web han podido aprovechar las mejoras que se han ido incorporando al lenguaje de marcado HTML para programar aplicaciones más funcionales y consistentes.

La extensión de los ficheros HTML suele ser *.html* o *.htm*.

Nota técnica

Para ver el código HTML de una página web hay que pulsar el botón derecho sobre esta y seleccionar la opción «Ver código fuente».

1.4.2. XML

XML (lenguaje de marcado extensible o *Extensible Markup Language*) es el lenguaje de intercambio de información entre sistemas por excelencia. Se puede afirmar que XML es para los sistemas el equivalente a HTML para los humanos. Un documento XML puede servir para almacenar la configuración de una aplicación, los mejores marcadores de un videojuego, el diseño de una interfaz de usuario o la información de un gráfico vectorial.

La primera versión de XML salió a la luz en 1998. Es, al igual que HTML, una especificación del W3C y se define como «de propósito general». Esto significa que es un metalenguaje al venir definido por reglas y no por etiquetas concretas. Tomando la especificación de XML se pueden definir dialectos específicos para una gran variedad de aplicaciones de todo tipo.

© Ediciones Paraninfo

Figura 1.7. *XML es un metalenguaje de marcas muy versátil y potente.*

La última versión vigente de XML en el momento de la escritura de este libro es la 1.1. Debido a su naturaleza flexible y no específica, XML no requiere de revisiones ni mejoras frecuentes, ya que alberga en sí mismo la capacidad suficiente para adaptarse a las necesidades presentes y futuras.

La extensión de los ficheros XML suele ser *.xml*.

Nota técnica

Las extensiones son un añadido que se agrega al nombre del fichero para facilitar su identificación, pero no obliga a que el contenido del fichero sea acorde a dicha extensión. Por lo tanto, cualquier extensión puede contener todo tipo de formato.

1.4.3. XHTML

El lenguaje de marcas XHTML *(eXtensible HyperText Markup Language)* se puede definir de manera simplificada como la versión en formato XML del lenguaje de marcas HTML. Se presentó en el año 2000 como una recomendación del W3C.

XHTML tiene la misma relación con XML que HTML con SGML. Tanto HTML como XHTML surgen a partir de un metalenguaje y ambos son lenguajes de marcado para el desarrollo de páginas web. La diferencia fundamental es que XHTML tiene una sintaxis más estricta y es más formal que HTML.

HTML tiene una serie de características que impiden que sea lo que se denomina **documento XML bien formado**, como por ejemplo la etiqueta *
*. Esta etiqueta indica al navegador web que tiene que mostrar un salto de línea y no tiene etiqueta de cierre, ya que no debe contener elementos en su interior. En XML todas las etiquetas de apertura deben cerrarse, por lo que el uso de la etiqueta *
* no sería aceptado. En la versión XML de HTML (XHTML) la etiqueta *
* se sustituye por *
*.

© Ediciones Paraninfo

Además de la obligatoriedad de cerrar todas las etiquetas, el documento XHTML debe cumplir con todas las características de un documento XML bien formado y esto implica ciertos cambios. Estos cambios no restan ni un ápice de capacidad de diseño y maquetación, ya que todas las posibilidades que proporciona HTML se mantienen en XHTML.

Figura 1.8. *XHML es la versión XML de HTML.*

El hecho de que XHTML sea un HTML «adaptado a XML» aparentemente no hace ninguna aportación a este último. Entonces ¿por qué emplear esfuerzo en crear un nuevo lenguaje de marcas para hacer lo mismo que uno que ya existe? Como cabe esperar, existen razones y estas tienen que ver con el éxito de HTML como lenguaje para la creación de páginas web.

HTML tiene una sintaxis definida por un estándar, tal y como se ha citado previamente, pero son los navegadores los programas que procesan los documentos y los representan en pantalla. Los navegadores, cuando encuentran un documento HTML mal construido (nombres de etiquetas incorrectas, atributos inexistentes, errores en la estructura del documento, etc.), interpretan («se inventan») lo que quería decir el autor de la página. La mayoría de las veces aciertan y este funcionamiento ha sido una gran contribución al éxito de la web: a más sencillez, más uso. Por otra parte, si la interpretación es errónea, el autor podrá ver en su navegador que la página está mal construida o, como mal mayor, la persona que acceda a la página verá una página mal maquetada.

Como consecuencia negativa, una página HTML mal construida es difícil de procesar por un programa informático, ya que estos no son tan flexibles como las personas. Si un programa «se inventa» qué es la cantidad de un producto en un listado y qué es el precio, las consecuencias pueden ser imprevisibles.

A continuación se muestra un ejemplo de documento XHTML.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
    "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>Título del documento</title>
```


© Ediciones Paraninfo

```
</head>
<body>
    <h1>Lenguajes de Marcas y Sistemas de Gestión de la Información</h1>
    <br />
    <h2>Unidad 1. Introducción</h2>
    <p>Texto de la unidad</p>
</body>
</html>
```

La estructura y la sintaxis son casi idénticas a las de un documento HTML, aunque llama la atención la etiqueta *
*. Esta etiqueta, en HTML, es *
*, pero sin el elemento de cierre el documento XHTML no estaría bien formado. En caso de utilizar *
* en el documento XHTML, el navegador mostrará un error advirtiendo que existe un desajuste entre la etiqueta de apertura y la final (Figura 1.9).

This page contains the following errors:

error on line 12 at column 9: Opening and ending tag mismatch: br line 0 and body

Below is a rendering of the page up to the first error.

Figura 1.9. *Mensaje de error mostrado por un navegador web al abrir un documento XHTML erróneo.*

Con XHTML es posible crear páginas web con el mismo etiquetado y la misma funcionalidad que con HTML, pero garantizando que la estructura y la sintaxis son correctas, ya que en este caso la validación es tan sencilla de realizar como con cualquier documento XML. Con XHTML los documentos son fácilmente procesables al cumplir estrictamente con la sintaxis de XML y los analizadores *(parsers)* son más sencillos de programar.

La extensión de los ficheros XHTML suele ser *.xhtml.*

1.4.4. JSON

JSON es el acrónimo de *JavaScript Object Notation* («notación de objetos de JavaScript»). En realidad, es el formato de texto en el que JavaScript representa sus objetos, incluidos los *arrays*. Pese a esa naturaleza original tan específica, JSON ha abandonado su vínculo con JavaScript y se ha universalizado como alternativa a XML para realizar almacenamiento e intercambio de datos entre sistemas independientemente de la plataforma y del lenguaje de programación. Casi todos los lenguajes que se usan en la actualidad disponen de analizadores JSON.

En el siguiente código se puede observar un ejemplo de fichero en formato JSON:

```
{
    "asunto": "Lenguaje de Marcas",
    "publicacion": "2021",
```

© Ediciones Paraninfo

```
    "editorial": "Paraninfo",
    "lenguajes": [
      {
        "nombre": "HTML",
        "uso": "Web"
      },
      {
        "nombre": "XML",
        "uso": "Intercambio de datos"
      },
      {
        "nombre": "JSON",
        "uso": "Intercambio de datos"
      }
    ]
}
```

JSON es más ligero y cuenta con una sintaxis más sencilla que XML, pero por el contrario no dispone de las herramientas y riquezas que sí tiene XML. El debate entre JSON y XML no debe existir: hay contextos en los que JSON es más apropiado que XML y viceversa. De hecho, es frecuente encontrar ambas alternativas conviviendo en un mismo sistema.

La extensión de los ficheros JSON suele ser *.json*.

1.4.5. SVG

SVG (*Scalable Vector Graphics* o gráficos vectoriales escalables) es un dialecto de XML. Se utiliza para representar gráficos vectoriales de dos dimensiones.

La extensión de los ficheros SVG suele ser *.svg* o *.svgz*.

1.4.6. MathML

MathML (*Mathematical Markup Language* o lenguaje de marcado matemático) es un dialecto de XML. Se utiliza para representar notaciones matemáticas.

1.4.7. NewsML

NewsML (*News Markup Language* o lenguaje de marcado de noticias) es un dialecto de XML. Se utiliza para representar noticias electrónicas.

1.4.8. FXML

FXML es un dialecto de XML. Se emplea para representar las interfaces de usuario en la tecnología JavaFX del lenguaje de programación Java.

© Ediciones Paraninfo

En el siguiente código se muestra un segmento de la definición de una interfaz gráfica utilizando este dialecto:

```
<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.scene.text.*?>
<?import javafx.scene.control.*?>
<?import java.lang.*?>
<?import javafx.scene.layout.*?>
<?import javafx.scene.layout.AnchorPane?>

<AnchorPane fx:id="paneSecurity" prefHeight="600.0" prefWidth="800.0"
xmlns="http://javafx.com/javafx/8" xmlns:fx="http://javafx.com/
fxml/1" fx:controller="com.lafabricadeandroides.websecfx.controller.
SecurityController">
```

1.4.9. XAML

XAML (*eXtensible Application Markup Language* o lenguaje de marcado extensible de aplicación) es un dialecto de XML. Es el lenguaje en el que se definen las interfaces de usuario en las aplicaciones para el sistema operativo Windows desarrolladas con la tecnología de Microsoft.

El siguiente código muestra la definición de parte de una interfaz gráfica desarrollada con XAML:

```
<?xml version="1.0" encoding="UTF-8"?>
<Button Content="Click me">
  <Button.Margin>
    <Thickness Left="10" Top="20" Right="10" Bottom="30"/>
  </Button.Margin>
</Button>
```

1.4.10. Android

En las aplicaciones Android, la forma habitual de diseñar las interfaces de usuario es mediante ficheros escritos en un dialecto de XML. En estos ficheros se definen los esquemas de distribución, así como los componentes y sus características que van a permitir al usuario interactuar con la aplicación.

El código que se muestra a continuación muestra un diseño simple de la interfaz de una aplicación Android:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
        android:layout_width="match_parent"
```

© Ediciones Paraninfo

```
    android:layout_height="match_parent"
    android:orientation="vertical" >
       <TextView android:id="@+id/text"
               android:layout_width="wrap_content"
          android:layout_height="wrap_content"
          android:text="Hello, I am a TextView" />
   <Button android:id="@+id/button"
          android:layout_width="wrap_content"
          android:layout_height="wrap_content"
          android:text="Hello, I am a Button" />
</LinearLayout>
```

1.4.11. Servicios web

Un servicio web *(web services)* es un componente software alojado en un servidor al que se accede utilizando protocolos estándares y tecnologías de la web como HTTP, HTTPS, XML o JSON.

Los denominados **servicios RESTful** (también llamados REST) utilizan con frecuencia dialectos de XML (en otras muchas ocasiones usan JSON) para el intercambio de información entre los dos puntos de la comunicación: el programa que proporciona los datos (el servicio web) y el que los consume (la aplicación que hace de cliente en la comunicación). Estos dialectos pueden ser alguno de los dialectos existentes (los citados en esta unidad y muchos más) o creados a medida para un caso concreto.

Figura 1.10. *Flujo de datos entre cliente y servidor durante la llamada a un servicio web.*

Por otra parte, los servicios web denominados **SOAP** *(Simple Object Access Protocol)* utilizan dialectos concretos de XML, ya que la especificación de este tipo de componente es mucho más precisa. Los formatos WSDL (*Web Service Description Language* o lenguaje de descripción de servicio web) o el utilizado para la creación de mensajes SOAP son algunos ejemplos.

1.4.12. RSS y Atom

Tanto RSS (*Really Simple Sindication* o sindicación realmente simple) como Atom son dialectos de XML. Se utilizan para la distribución de contenidos de fuentes a los que los usuarios se han suscrito previamente, siendo Atom una alternativa a RSS creado con posterioridad.

© Ediciones Paraninfo

El siguiente bloque de código muestra un segmento del documento de avisos de fenómenos meteorológicos de la Agencia Estatal de Meteorología (AEMET) en formato RSS:

```
<?xml version="1.0" encoding="UTF-8"?>
<rss version="2.0">
  <channel>
    <title>AEMET - Avisos de fenómenos meteorológicos adversos -
Extremadura</title>
    <link>http://www.aemet.es/es/eltiempo/prediccion/avisos?k=ext</link>
    <description>Avisos de fenómenos meteorológicos adversos según el Plan
Meteoalerta de la Agencia Estatal de Meteorología (AEMET) para Extremadura.</
description>
    <language>es</language>
```

1.4.13. XSL

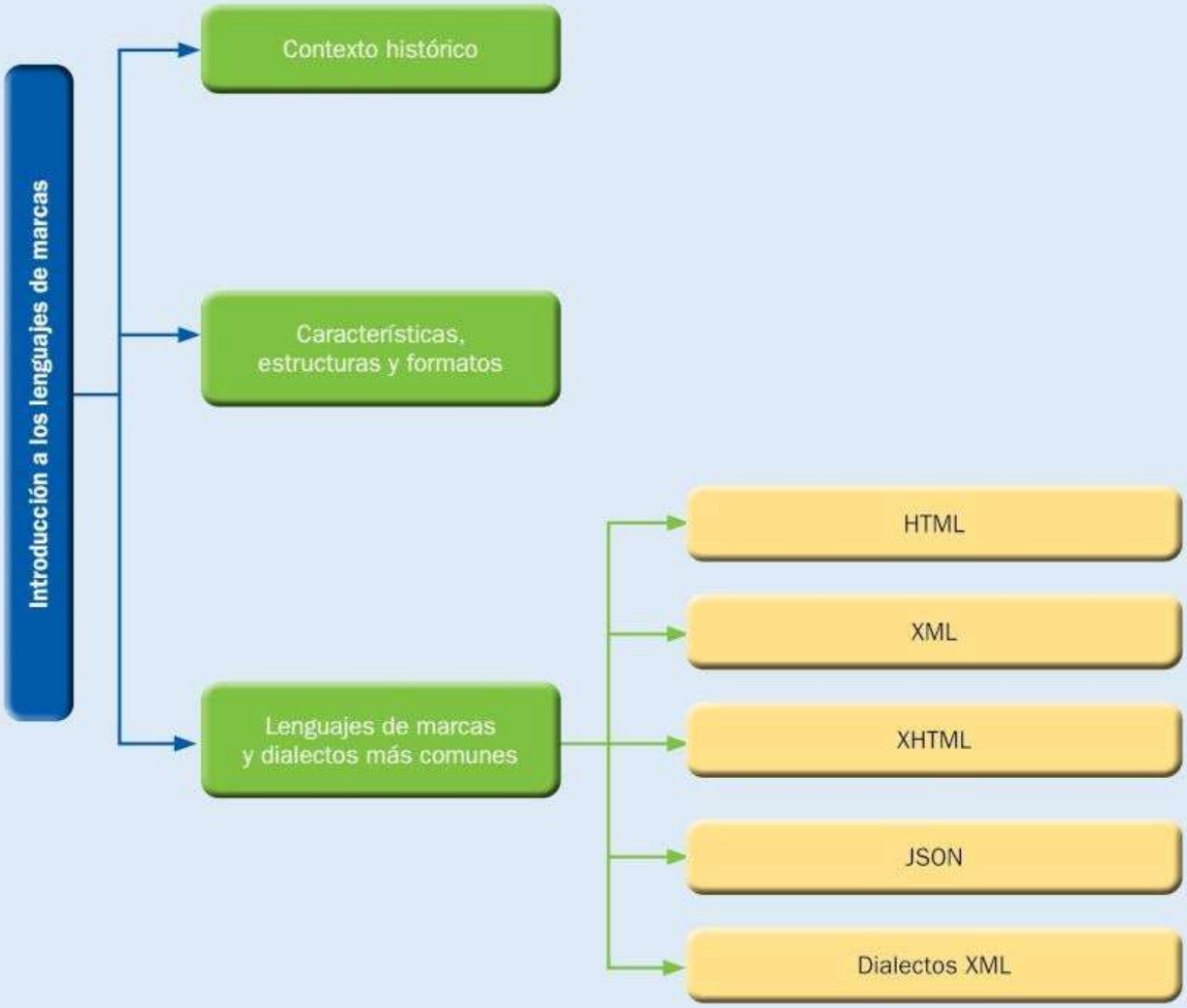
XSL (*eXtensible Stylesheet Language* o lenguaje de hojas de estilo extensible) es una especificación del W3C utilizada para transformar y manipular documentos XML. Permite indicar cómo ha de presentarse la información contenida en un documento XML.

XSL consta de varias recomendaciones y varias de ellas se definen mediante documentos escritos en formato XML.

Nota técnica

Existen muchos más lenguajes de marcas que los citados en esta unidad.

© Ediciones Paraninfo



© Ediciones Paraninfo

Actividades de comprobación

1.1. Indica la característica más destacable de los lenguajes de marcas:

a) No son lenguajes de programación.
b) Permiten representar información.
c) Se basan en marcas o etiquetas.
d) Todas las respuestas son correctas.

1.2. El ejemplo más representativo de lenguaje de marcas orientado a la presentación de información es:

a) GML.
b) SGML.
c) HTML.
d) XML.

1.3. HTML es un lenguaje de marcas creado a partir de:

a) SGML.
b) XML.
c) XHTML.
d) SVG.

1.4. XML es un lenguaje de marcas derivado de:

a) SGML.
b) XML.
c) XHTML.
d) SVG.

1.5. Indica cuál de las siguientes opciones es una de las características fundamentales de los lenguajes de marcas:

a) Dependen de la plataforma.
b) Se almacenan en formatos complejos codificados.
c) Son flexibles.
d) Son difíciles de procesar.

1.6. Señala cuál de las siguientes opciones es una de las características fundamentales de los lenguajes de marcas:

a) Es independiente de la plataforma.
b) Se almacenan en formatos complejos codificados.
c) Son rígidos.
d) Son difíciles de procesar.

1.7. XHTML es un lenguaje derivado de:

a) SGML.
b) XML.
c) HTML.
d) Todas las respuestas son correctas.

© Ediciones Paraninfo

1.8. ¿Cuál de los siguientes dialectos de XML se utiliza para representar gráficos?

a) SVG.
b) MathML.
c) NewsML.
d) FXML.

1.9. De los siguientes dialectos de XML, ¿cuál se emplea para representar funciones matemáticas?

a) SVG.
b) MathML.
c) NewsML.
d) FXML.

1.10. ¿Qué lenguaje de marcas se utiliza en el diseño de los servicios web?

a) XML.
b) XHTML.
c) HTML.
d) FXML.

Actividades de aplicación

1.11. Explica en qué se diferencian HTML y XHTML.

1.12. Enumera al menos tres de las características de los lenguajes de marcas.

1.13. Elabora un resumen con la evolución cronológica de los lenguajes de marcas hasta llegar a HTML.

1.14. Explica brevemente las diferencias entre un lenguaje de marcas orientado a la presentación de otro orientado al almacenamiento y procesado de la información.

1.15. Diseña un nuevo lenguaje de marcas. Debe permitir marcar un texto como importante, de color verde y escrito en idioma inglés.

1.16. Entra en la página web del ayuntamiento de un municipio cualquiera, consulta el código fuente y busca un bloque de código HTML que incumpla con las normas de XML. Indica qué norma o normas incumple.

1.17. Explica brevemente qué es un dialecto de XML.

1.18. Busca y enumera tres dialectos de XML que no conozcas hasta ahora, indicando para qué se utilizan.

1.19. Halla en internet algún ejemplo de servicio web que proporcione datos en formato XML.

1.20. Busca en internet un servicio web que proporcione datos en formato JSON sobre películas de cine.

1.21. Localiza y descarga un fichero XML de internet. Modifícalo para que contenga algún error. Abre el fichero con un navegador web y comprueba que el mensaje y la posición del error son correctos.

© Ediciones Paraninfo

1.22. Busca en la web una fórmula expresada en MathML. Elabora un listado con al menos tres de las etiquetas que utiliza incluyendo su significado.

Actividades de ampliación

1.23. En la web http://web.archive.org/ existen copias de las páginas web a lo largo del tiempo. Compara el aspecto actual de al menos dos páginas web con las versiones de años atrás.

1.24. Lee información sobre Tim Berners-Lee, creador de la web. Redacta una ficha con fechas y detalles de su trayectoria profesional.

1.25. Busca información sobre la historia de internet. Elabora un resumen el que se explique quién lo creó y con qué objetivo, e indica las fechas más relevantes.

1.26. Localiza alguna captura de pantalla del navegador Mosaic, uno de los primeros de la historia. Comenta las diferencias con los navegadores actuales.

1.27. Encuentra el código fuente de la primera página web del mundo. Comprueba si alguna de las primeras cinco etiquetas que se usan en dicha página está obsoleta.

1.28. Descarga un fichero en formato JSON y ábrelo con Google Chrome y con Mozilla Firefox. Enumera las diferencias de representación en ambos navegadores.

Enlaces web de interés

- **w3c.es** - https://www.w3c.es
  (Portal web en español del Consorcio World Wide Web)
- **Tim Bernes-Lee** - https://www.w3.org/People/Berners-Lee/
  (Breve biografía sobre el inventor de la World Wide Web)
- **Primer sitio web** - http://info.cern.ch/
  (Información sobre el primer sitio web de la historia)
- **Chrome** - https://www.google.com/intl/es_es/chrome/
  (Portal web del navegador Google Chrome)
- **Firefox** - https://www.mozilla.org/es-ES/firefox/new/
  (Portal web del navegador Mozilla Firefox)

© Ediciones Paraninfo

- **Opera** - https://www.opera.com/es
  (Portal web del navegador Opera)

- **Edge** - https://www.microsoft.com/es-es/edge
  (Portal web del navegador Microsoft Edge)

- **Safari** - https://www.apple.com/es/safari/
  (Portal web del navegador de los ordenadores y dispositivos Apple Safari)

- **Internet Archive** - https://web.archive.org/
  (Capturas de páginas web a través del tiempo)

© Ediciones Paraninfo

UNIDAD 2

Los lenguajes de la web. HTML y XHTML

Objetivos

- Descubrir la historia y la evolución de HTML.
- Presentar la estructura y la sintaxis de los documentos HTML.
- Conocer los elementos básicos del lenguaje.
- Profundizar en el conocimiento de las distintas etiquetas avanzadas, su significado y uso.
- Aprender a construir documentos basados en etiquetado semántico.
- Desarrollar los mecanismos de maquetación y el comportamiento de los tipos de elementos.
- Comparar el lenguaje HTML con XHTML, su equivalente en lenguaje XML.
- Enumerar algunas de las herramientas disponibles para visualizar y editar documentos HTML y XHTML.

Contenidos

- 2.1. Introducción, evolución y estado actual de HTML
- 2.2. Estructura y sintaxis del lenguaje HTML
- 2.3. Elementos de HTML
- 2.4. Tablas en HTML: estructura y elementos
- 2.5. Formularios en HTML: estructura y elementos
- 2.6. Multimedia en HTML
- 2.7. XHTML: la versión XML de HTML
- 2.8. Herramientas de consulta y edición de HTML

Introducción

La *World Wide Web,* más conocida como web, es el servicio de la red internet más popular. Su principal objetivo es permitir la distribución de documentos entre personas, con independencia de la plataforma o del sistema operativo que utilicen. La web es, junto con el correo electrónico, uno de los servicios que han hecho de internet la red global que es.

HTML (junto con XHTML, su versión XML) es el lenguaje que se utiliza para dar forma a los documentos publicados en la web. Es gratuito, sencillo, potente y no requiere de ningún software sofisticado para su edición, ya que un editor de texto es suficiente para crear cualquier tipo de documento.

2.1. Introducción, evolución y estado actual de HTML

HTML es, sin ninguna duda, el lenguaje de marcas más conocido y utilizado. Sus siglas significan en castellano «lenguaje de marcado de hipertexto» *(HyperText Markup Language)* y es el lenguaje en el que están escritas la gran mayoría de páginas de la *World Wide Web* (lo que comúnmente se conoce como web). Obviamente, todos los navegadores web reconocen HTML y lo saben interpretar y traducir, generando la representación de la página web tal y como la vemos los usuarios. El lenguaje HTML y la web son términos que van necesariamente de la mano, ya que de manera independiente no habrían tenido la repercusión que han conseguido conjuntamente.

En la actualidad, HTML está gestionado por el W3C (World Wide Web Consortium), un consorcio dedicado a la estandarización de la web para lograr que esta sea accesible, universal, fiable, fácil de usar y de uso libre. **HTML es, por lo tanto, gratuito.** Ninguna persona ni empresa debe pagar por utilizarlo.

Nota técnica

La web es uno de los servicios de internet, pero no el único; el correo electrónico y la transferencia de archivos son algunos de los más utilizados. Existen muchos más servicios, algunos en desuso, como Gopher o Telnet.

En 1991 Tim Berners-Lee diseñó HTML como el lenguaje de marcado capaz de proporcionar un mecanismo sencillo, flexible y universal para almacenar y transmitir información entre distintos sistemas informáticos. Aunque las redes de ordenadores ya existían desde muchos años antes, internet proponía una red global de comunicaciones y había que dotarla de soluciones para que las personas que las fuesen a utilizar pudiesen compartir información fácilmente. Ese primer diseño era muy simple e incluía 18 etiquetas, algunas de las cuales aún se mantienen en la versión 5 de HTML. Difícilmente Tim Berners-Lee pudo anticipar el éxito de su creación, lo cual no le quita ningún mérito. HTML está bien diseñado, es sólido y se fundamenta en unas bases acertadas.

© Ediciones Paraninfo

Existen dos hechos fundamentales que han marcado el destino de HTML:

- Los destinatarios de HTML han sido, desde el principio, **las personas.**
- La información proporcionada por HTML es de **naturaleza estética** y no semántica.

Estas dos características tienen algunas consecuencias. Una de ellas es que los navegadores, al procesar el código HTML, si encuentran algún error de sintaxis, pueden intentar solucionarlo, y si la solución propuesta no es correcta, los resultados no serán catastróficos. Las personas tenemos una capacidad de interpretación de la información mucho más flexible que las máquinas, por lo que, ante un error de interpretación por parte del navegador, el resultado nos resultará inconsistente o poco atractivo estéticamente, pero seguro que seremos capaces de interpretarlo de forma correcta. La semántica la aporta el usuario en función de lo que ve y de su buen criterio. Si la página no está bien maquetada, el autor de esta se dará cuenta visualmente del error. Si no lo percibe, es que el error no es relevante y las consecuencias no serán graves.

Nota técnica

Para abrir con una navegador web un documento HTML almacenado en el sistema de archivos local en el sistema operativo Windows, hay que pulsar con el botón derecho del ratón sobre el fichero, seleccionar la opción **Abrir con** y elegir el navegador deseado.

Por otro lado, si es un programa informático el que tiene que analizar y procesar el texto, este ha de ser riguroso y carente de errores sintácticos. Un programa de ordenador es determinista (ante las mismas entradas debe generar las mismas salidas) y esto implica que no dispone de la capacidad de **interpretar** en el sentido humano.

Es por estas circunstancias por las que los navegadores web son laxos en la interpretación del código HTML y es una de las razones de la popularidad y alcance de la web. Si hacer páginas HTML fuese muy difícil, su aceptación habría sido probablemente más lenta. Por suerte, los diseñadores y programadores de los navegadores entendieron que el proceso de escribir una página web no puede ser igual de estricto que el proceso de escribir un programa informático en cualquier lenguaje de programación.

HTML, desde su nacimiento, ha pasado por diversas revisiones que han ido ampliando y mejorando la capacidad del lenguaje.

En la Tabla 2.1 se muestran las diferentes versiones de HTML y su año de publicación.

Tabla 2.1. Evolución de las versiones de HTML

Año	Versión	Observaciones
1991	Diseño inicial	Incluye 18 etiquetas. Se considera como un dialecto de SGML. Se plasma en un documento interno e informal del CERN (Conseil Européen pour la Recherche Nucléaire u Organización Europea para la Investigación Nuclear, en español).
1992	HTML 1.1	Primer borrador informal.

© Ediciones Paraninfo

Año	Versión	Observaciones
1995	HTML 2.0	Publicado por el IETF (Internet Engineering Task Force o Grupo de Trabajo de Ingeniería de Internet, en español).
1997	HTML 3.2	Publicado como recomendación del W3C (World Wide Web Consortium).
1997	HTML 4.0	Publicado como recomendación del W3C.
1999	HTML 4.1	Publicado como recomendación del W3C.
2000	XHTML	Publicado como recomendación del W3C. Tenía como objetivo sustituir a HTML. En HTML5 se incorpora como parte de la recomendación y deja de desarrollarse por separado.
2014	HTML 5.0	Publicado como recomendación del W3C.
2016	HTML 5.1	Publicado como recomendación del W3C.
2017	HTML 5.2	Publicado como recomendación del W3C.

Cada versión mejora a la anterior. Incorpora nuevas etiquetas, elimina otras obsoletas y se acomoda a las capacidades tecnológicas. Hay que tener en cuenta que el avance en las tecnologías relacionadas con la informática y las telecomunicaciones es de gran importancia y se da en cortos periodos de tiempo.

Recuerda

Tanto HTML como el resto de las tecnologías relacionadas con internet están vivas y en constante evolución. La versión actual cambiará en el futuro, puliendo defectos y ampliando capacidades. Las profesiones relacionadas con la tecnología exigen de un aprendizaje constante.

Por ejemplo, en los primeros años de existencia de internet, el acceso a la red se hacía a través de la línea telefónica analógica mediante el uso de un módem (conversor de señales digitales-analógicas y analógicas-digitales) a una velocidad impensable para las velocidades de las líneas digitales actuales. Se ha pasado en treinta años de limitar el número de bytes que tiene una fotografía para agilizar la carga de la página a retransmitir vídeo en *streaming* en alta definición.

Figura 2.1. *El incremento de la velocidad de las comunicaciones ha permitido mejorar HTML.*

© Ediciones Paraninfo

HTML ha ido adaptándose con éxito a esos avances para seguir siendo el lenguaje estándar de marcado para la difusión de contenidos a través de la web, aceptado por toda la industria tecnológica sin objeciones.

La versión 5 (y posteriores) de HTML ha recogido las necesidades de un mercado dinámico y vertiginoso y ha abierto un nuevo e importante conjunto de oportunidades gracias a su potencial y a su capacidad, respetando los principios de su fundación.

Actividad propuesta 2.1

Servicios de internet

Busca información referente a los siguientes servicios de internet:

- Servicio de transferencia de archivos FTP *(File Transfer Protocol).*
- Servicio de acceso a información Gopher.
- Servicio de acceso remoto a máquinas Telnet.

2.2. Estructura y sintaxis del lenguaje HTML

El lenguaje de marcado HTML define el contenido de una página web. Los navegadores interpretan las marcas contenidas en los documentos HTML y representan la información para que los usuarios la puedan consultar e interactuar con ella. Trabajando junto con las hojas de estilo CSS, HTML es un increíble lenguaje de representación de información.

El lenguaje HTML ha tenido un enorme éxito debido, en parte, a su sencilla estructura y sintaxis.

2.2.1. Estructura y sintaxis

Los elementos que forman las páginas HTML están identificados por marcas o etiquetas. Estas marcas «envuelven» el contenido de la página, ya sea este texto, imágenes o cualquier otro tipo de elemento. Las marcas o etiquetas están compuestas por un nombre rodeado de los símbolos «<» y «>».

```
<etiqueta>
```

El nombre de la etiqueta puede estar escrito en mayúsculas o minúsculas, ya que no se distingue entre ambos tipos de letra. No obstante, se recomienda que el etiquetado sea uniforme.

Existen dos tipos principales de etiquetas: las etiquetas de apertura y las de cierre. Un contenido debe estar rodeado por ambas etiquetas. La etiqueta de apertura se incluye antes del contenido y la etiqueta de cierre después de este. La etiqueta de apertura se diferencia de la etiqueta de cierre en que esta última incluye el símbolo «/» justo antes del nombre.

```
</etiqueta>
```

© Ediciones Paraninfo

Ambas etiquetas tendrán necesariamente el mismo nombre y rodearán un contenido. El conjunto formado por el **contenido** y las etiquetas de **apertura** y **cierre** forman un **elemento.**

```
<etiqueta>Esto es el contenido</etiqueta>
```

En HTML no todas las etiquetas necesitan apertura y cierre, ya que no todas son contenedoras. Un ejemplo es la etiqueta «
». Esta etiqueta provoca un salto de línea en el texto, pero no debe rodear contenido, ya que el salto de línea es independiente del contexto. Por lo tanto, algunos elementos solo estarán formados por una etiqueta de apertura. A estos elementos se les denomina **elementos vacíos**.

```
<etiquetasincierre>
```

Dentro de un elemento se pueden incluir atributos. Los atributos permiten caracterizar el elemento de cara a identificarlo, representarlo o tratarlo de manera específica. La existencia de atributos es opcional y su número puede variar. Todos los elementos de un documento HTML pueden disponer de atributos, pero no todos los atributos son aplicables a cualquier elemento.

Los atributos se incluyen en la etiqueta de apertura y están compuestos por un nombre, el símbolo «=» y el valor del atributo entrecomillados. Las comillas pueden ser simples o dobles. De nuevo, se recomienda que el uso de comillas simples o dobles sea homogéneo a lo largo del documento.

```
<etiqueta atributo1="valor1" atributo2="valor2">
```

Los elementos de un documento HTML se pueden anidar. Un elemento puede contener a uno o más elementos en su interior.

```
<elementonivel1>
        <elementonivel2>
                <elementonivel3>
                        Contenido
                </elementonivel3>
        </elementonivel2>
</elementonivel1>
```

Esta anidación debe estar bien construida. No se permite que los elementos se solapen. En el siguiente código se muestra un ejemplo de una anidación mal construida:

```
<elementonivel1>
        <elementonivel2>
                <elementonivel3>
                        Contenido
        </elementonivel2>
                </elementonivel3>
</elementonivel1>
```


© Ediciones Paraninfo

Una página HTML está compuesta por un conjunto de elementos y alguna información adicional que se indicará a continuación. Este conjunto de elementos debe respetar una estructura definida en el estándar de HTML.

En el siguiente código se muestra la estructura básica de una página HTML:

```
1  <!DOCTYPE html>
2  <html>
3  <head>
4
5  </head>
6  <body>
7
8  </body>
9  </html>
```

El primer elemento (línea 1) identifica el tipo del documento. Este elemento es un elemento vacío y, por lo tanto, no requiere de etiqueta de cierre. Aunque no es exactamente un elemento de HTML se incluye en las páginas HTML para indicar al navegador de qué tipo es el documento que va a procesar.

El siguiente elemento (entre las líneas 2 y 9) se define con la etiqueta <html>. Es el elemento raíz y dentro de él se encuentra todo el contenido de la página.

El elemento etiquetado como <head> (entre las líneas 3 y 5) incluye información no visible en la representación de la página dentro del navegador pero que sí afecta a cómo esta va a ser presentada. En esta sección se indica, por ejemplo, el título que aparece en la pestaña del navegador, la hoja de estilos CSS que utilizar o el juego de caracteres que se usará para la representación de los textos.

Por último, el elemento <body> (entre las líneas 6 y 8) albergará el contenido visible de la página, ya sea este texto, tablas, listas, imágenes o elementos multimedia, por citar algunos ejemplos.

Actividad resuelta 2.1

Creación de documento HTML básico

Crear una página HTML con la estructura básica (elementos <html>, <head> indicando el juego de caracteres y el título y <body> conteniendo un texto plano. Utilizar el programa *Bloc de Notas* de Windows para escribirlo.

Solución

- Crear un fichero con el nombre ejercicio-resuelto-2-1.html.
- Abrir el fichero con el *Bloc de Notas* de Windows y escribir el siguiente código:

```
<!DOCTYPE html>
<html lang="es">
<head>
   <meta charset="UTF-8">
   <title>Documento básico</title>
```

© Ediciones Paraninfo

```
</head>
<body>
    Este es un documento básico.
</body>
</html>
```

- Guardar el fichero y abrir con un navegador web.

2.2.2. Entidades

Algunos caracteres tienen un significado especial en HTML, son invisibles o, sencillamente, no existen en el lenguaje en el que está escrita la página. Por ejemplo, el símbolo «menor que» (<) es el comienzo de una etiqueta, los espacios en blanco múltiples son ignorados o las tildes no existen en algunos idiomas. Las entidades permiten indicar al navegador que debe representar un carácter determinado, solucionando los problemas derivados de estas circunstancias.

Por ejemplo, utilizando el juego de caracteres ISO-8859-1, el texto **¡Soy un campeón!** se muestra en el navegador como **Â¡Soy un campeÃ³n!.** Como se puede observar, el símbolo «¡» y el carácter «ó» no se han visualizado correctamente. Aunque este problema se podría resolver utilizando el juego de caracteres «UTF-8», también se podría solucionar sustituyendo los caracteres problemáticos por su entidad equivalente: el carácter «¡» se sustituye por la entidad **¡** y el carácter «ó» por **ó**. El texto en el documento HTML quedaría de la siguiente manera: **¡Soy un campeón!**

Otro ejemplo tiene que ver con cómo se interpretan los espacios múltiples en HTML, ya que solo se respeta el primero de ellos. Según este comportamiento, el texto **Dejemos espacio entre la A y la B:A B** se muestra en el navegador como **Dejemos espacio entre la A y la B:A B.** Se observa que no se han respetado los espacios en blanco múltiples entre las letras A y B, quedándose los cinco espacios del código HTML original en uno único mostrado en el navegador. La entidad ** ** representa un espacio en blanco, por lo que la solución consistiría en sustituir el código HTML original por **Dejemos espacio entre la A y la B:A   B.**

Una entidad en HTML es un conjunto de caracteres que permiten indicar al navegador que represente caracteres que, por diversas razones, no se representan correctamente. Comienzan por el símbolo «&» y terminan con el símbolo «;».

Algunas de las entidades más comunes se pueden consultar en la Tabla 2.2.

Tabla 2.2. Las entidades resuelven problemas de representación

Carácter	Entidad
&	&
<	<
>	>

© Ediciones Paraninfo

Carácter	Entidad
“	"
‘	'
€	€
©	©

Para saber más

Se puede consultar la lista completa de entidades en la lista oficial:
https://html.spec.whatwg.org/multipage/named-characters.html#named-character-references

Actividad propuesta 2.2

Entidades

Busca las entidades correspondientes a los siguientes caracteres y símbolos en la lista oficial:

- $
- £
- ∞
- ®

2.2.3. Comentarios

HTML admite incluir **líneas de comentarios.** Estos elementos no tienen ningún efecto sobre la página, ya que su único objetivo es proporcionar información a los diseñadores y programadores cuando están consultando el código interno de esta. El contenido se delimita con los caracteres `<!--` y `-->` y será ignorado por el navegador. A continuación, se muestra un ejemplo de un comentario incrustado en un bloque de código HTML:

```
<article id="capitulo1">
    <!-- CONTENIDO DEL CAPÍTULO 1-->
</article>
```

Actividad propuesta 2.3

Construcción de una página HTML básica

Escribe en un editor de texto sencillo (el *Bloc de Notas* de Windows, por ejemplo) el siguiente código:

```
<!DOCTYPE html>
<html lang="es">
```

© Ediciones Paraninfo

```
<head>
    <meta charset="UTF-8">
    <title>Mi primera página</title>
</head>
<body>
    <!--Esta es mi primera página HTML-->
    <h1>¡HOLA!</h1>
</body>
</html>
```

Guarda el fichero con el nombre **mipagina.html**

Abre el fichero con un navegador web.

Prueba con varios navegadores: Google Chrome, Firefox, Opera, Microsoft Edge y Safari si tienes acceso a un ordenador de la marca Apple.

2.3. Elementos de HTML

Un documento HTML está compuesto de elementos organizados de manera jerárquica. El elemento <html> es la raíz y todos los demás descienden directa o indirectamente de él.

Como se ha indicado anteriormente, los elementos <html>, <head> y <body> son fundamentales dentro de la construcción de la página para indicar que el documento es HTML (elemento <html>), cómo se ha de representar la información (elemento <head>) y qué información se debe representar (elemento <body>), pero solo son la punta de un enorme iceberg.

La relación de elementos de HTML es muy extensa y admite múltiples clasificaciones. En esta sección se muestran los elementos organizados por familias en función de su funcionalidad. En algunos casos, debido a su singularidad y relevancia, se muestran elementos de manera individual y no agrupados. Es necesario indicar que la relación no es exhaustiva, ya que el lenguaje es muy amplio y el objetivo de este libro es ser didáctico y no un manual de referencia.

Cada elemento se identifica por una etiqueta y se caracteriza por un conjunto de atributos, muchos de los cuales son específicos del elemento en concreto. No obstante, existen algunos atributos globales que es importante conocer, ya que son determinantes para el correcto funcionamiento de tecnologías relacionadas con HTML como las hojas de estilo CSS o el lenguaje de programación JavaScript. Los atributos globales son comunes a todos los elementos HTML.

Para saber más

En la actualidad, el lenguaje utilizado para programas embebidos en los documentos HTML es JavaScript, un dialecto de la especificación ECMAScript. No obstante, en los primeros años de la web, existían más lenguajes que competían entre sí, además de JavaScript, como Visual Basic Script (VBScript) de Microsoft y JScript, una implementación de ECMAScript también de Microsoft.

© Ediciones Paraninfo

En la Tabla 2.3 se detallan algunos de los atributos globales más utilizados.

Tabla 2.3. Los atributos globales son compartidos por todos los elementos

Atributo	Descripción
class	Permite agrupar elementos. El valor de este atributo es una cadena de caracteres definida por el autor de la página. Si varios elementos comparten el mismo valor, se dice que son de la misma clase. Se utiliza para asignar estilos y durante la programación con JavaScript.
contenteditable	Si existe, el contenido del elemento es editable.
hidden	Si existe, el elemento no se representa en el navegador.
id	Identifica al elemento de manera única en el documento. El valor es una cadena de caracteres definida por el autor de la página. Se utiliza para asignar estilos y durante la programación con JavaScript.
lang	Determina el idioma en el que está escrito el contenido del elemento.
spellcheck	Determina si el elemento debe ser analizado ortográfica y gramaticalmente por el navegador.
style	Permite asignar un estilo al componente.
tabindex	Determina el orden de selección de elementos cuando se pulsa el tabulador.
title	Permite especificar información extra al elemento.
translate	Determina si el elemento debe ser traducido o no.

En el siguiente código de ejemplo se pueden ver algunos de los atributos globales de HTML:

```
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    <h2 id="encabezado" class="rojo">Encabezado</h2>
    <p class="rojo" style="background-color: aquamarine;">
        Este párrafo será visible en el navegador
    </p>
    <p id="explicacion" hidden>
        Este párrafo no será visible en el navegador
    </p>
    <p contenteditable spellcheck="true" tabindex="2">
        Este parrafo tiene faltas de ortografia y es editable
    </p>
```

© Ediciones Paraninfo

```
    <p lang="en" translate="no">
        This paragraph is written in English
    </p>
    <input type="text" tabindex="1">
    <input type="text" tabindex="3">
</body>
</html>
```

La representación generada por el navegador es la siguiente:

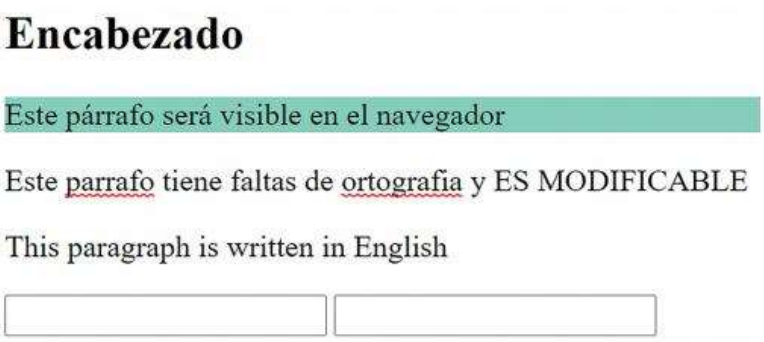


Figura 2.2. *Los atributos globales se pueden utilizar en todos los elementos de HTML.*

Algunas de las características que proporcionan los atributos globales no son visibles. Por ejemplo, el párrafo que tiene el atributo **hidden** no se muestra en el navegador. Tampoco la navegación con las pulsaciones de la tecla de tabulación: la primera pulsación posiciona el cursor en el primer elemento **<input>**; la segunda pulsación posiciona el cursor en el párrafo que tiene el atributo **contenteditable**; la tercera pulsación posiciona el cursor en el último elemento **<input>**. También hay atributos globales que tienen un efecto visible en la página: el color de fondo o el resultado de realizar la corrección ortográfica (depende del navegador). El resto de los atributos globales no tienen un efecto visible ni interno *a priori,* como el atributo **id** y el atributo **class**, que sirven para poder referenciar a los elementos desde las hojas de estilo CSS y desde programas escritos en JavaScript.

Nota técnica

Existen elementos como, por ejemplo, **<tt>** o ** ** cuyo uso está desaconsejado en HTML5, ya que están obsoletos. El mismo resultado que producen estos elementos se puede obtener utilizando estilos CSS y la recomendación es hacerlo de esta manera. Estos elementos no se incluyen en este libro para no inducir a confusión.

2.3.1. Elemento html

Es el elemento principal y la raíz del documento HTML. Todo el contenido del documento HTML, agrupado en los elementos <head> y <body>, está incluido dentro de este elemento.

© Ediciones Paraninfo

2.3.2. Elemento head

Es la primera de las secciones de una página web. No será directamente visible en la representación del documento ni en el navegador propiamente dicho (con alguna excepción), pero sí tendrá efecto directo y relevante en dicha representación.

El elemento más habitual que se incluye en esta sección es <title>. Contiene el título de la página, un texto que se muestra en la pestaña del navegador web.

```
<title>Título del documento</title>
```

El navegador (en este caso Mozilla Firefox) mostrará el título tal y como se refleja en la Figura 2.3.

Figura 2.3. *El título de la pestaña mejora la navegación y el posicionamiento de la página.*

Además de mostrarse en la pestaña correspondiente del navegador, el título es el nombre que se asigna a una dirección web cuando se almacena como favorito en el navegador. El título también lo tienen en cuenta los buscadores de internet para clasificar la página, por lo que una correcta asignación de este hará que el posicionamiento sea más acorde al contenido de dicha página.

El elemento «title» no es obligatorio, pero sí altamente recomendable, ya que en caso de no utilizarse el navegador, mostrará el nombre del fichero y esto, de cara al usuario final, es una información irrelevante y probablemente confusa.

Nota técnica

La optimización del posicionamiento en buscadores, conocido como SEO *(Search Engine Optimization)* está formada por un conjunto de técnicas que pretenden mejorar la posición en la que aparecerá una página web al hacer una búsqueda relacionada en un buscador como Google. La etiqueta **<title>** es uno de los elementos que utilizan los buscadores para posicionar la página.

Además del título, en esta sección se especifican los denominados **metadatos.** Un metadato es, en este contexto y utilizando una definición muy simplificada, un dato descriptivo que va a afectar a cómo se interpretan los datos contenidos en el documento.

Los metadatos se identifican con la etiqueta <meta> seguida de los atributos y valores correspondientes a la información que se quiere proporcionar al navegador.

© Ediciones Paraninfo

En la Tabla 2.4 se indican algunos de los metadatos más habituales.

Tabla 2.4. Los metadatos definen características generales del documento HTML

Metadato	Descripción
charset	Codificación del juego de caracteres. El valor estándar utilizado en HTML5 es UTF-8. Con anterioridad a esta versión de HTML el valor estándar era ISO-8859-1, por lo que es frecuente encontrar esta codificación en muchos documentos HTML.
http-equiv	Se utiliza para especificar una directiva. Todos los valores permitidos son nombres de encabezados HTTP.
name	Nombre del metadato. Se debe utilizar juntamente con el metadato **content.**
content	El valor asociado a atributos como **name** y **http-equiv.** Dependiendo del tipo de atributo, será un texto libre o una opción dentro de un conjunto.

Los valores más habituales del metadato «charset» son UFT-8 (codificación Unicode) e ISO-8859-1 (codificación del alfabeto latino).

```
<meta charset="UTF-8">
```

Nota técnica

La lista completa de juegos de caracteres que se pueden utilizar referentes al metadato «charset» se encuentra publicada en la web de la Internet Assigned Numbers Authority (IANA), que es la entidad que se encarga de supervisar asignaciones de recursos de internet tales como las direcciones IP.

Se puede consultar la lista completa de juegos de caracteres aceptados en el enlace siguiente:

http://www.iana.org/assignments/character-sets/character-sets.xhtml

Por su parte, el metadato «http-equiv» tiene más alcance, por lo que la definición es sustancialmente más compleja. Mediante este metadato se pueden especificar aspectos tan diferentes como el tipo de caché que requiere la página, qué debe hacer el navegador ante la descarga de determinado documento, si el contenido está codificado con algún algoritmo de compresión, la fecha de publicación o cada cuanto tiempo debería recargar la página el navegador de manera automática.

En el siguiente listado se muestran algunos ejemplos de uso de este metadato junto con la descripción de su comportamiento.

- «**Content-type**». Especifica el tipo de contenido y el juego de caracteres. En el ejemplo que se muestra a continuación es de tipo texto utilizando el lenguaje html y la codificación UTF-8.

```
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
```


© Ediciones Paraninfo

- **«cache-control»**. Especifica cómo ha de gestionar el navegador la página HTML desde el punto de vista de la caché. Se puede indicar una fecha de expiración, que solo se almacene en memoria y no en soporte físico o desactivar por completo el almacenamiento en caché con el objetivo de que cada vez que se consulte la página se vuelva a descargar al servidor. En el siguiente ejemplo se está indicando al navegador que no almacene la página en caché.

```
<meta http-equiv="cache-control" content="no-store">
```

- **«refresh»**. Indica al navegador que, pasado un determinado periodo de tiempo, refresque la página o, alternativamente, cargue una página referenciada a través de una URL. En la etiqueta que se muestra como ejemplo a continuación se está indicando al navegador que, pasados tres segundos, cargue la página principal del sitio web de Ediciones Paraninfo.

```
<meta http-equiv="refresh" content="3; url=http://www.paraninfo.es/">
```

Los posibles valores de los atributos «name» y «content» son muy diversos y hacen referencia a una gran variedad de características que se pueden configurar para determinar el comportamiento de la página y del navegador. A continuación, se muestran algunos de ellos:

- **«robots»**. Hace recomendaciones a los robots utilizados por los motores de búsqueda. Permite indicar, por ejemplo, que no se desea que se indexe una página o las páginas a las que dirigen los enlaces de la página actual. También se puede hacer saber al robot que se quiere que la página sea indexada, aunque esto lo suelen hacer por defecto. En el ejemplo siguiente se indica a los potenciales robots que no indexen la página:

```
<meta name="robots" content="noindex"/>
```

- **«keywords»**. Indica palabras clave que servirán para que los indexadores de los motores de búsqueda sean capaces de clasificar de una manera más eficiente la página. El número de palabras clave puede ser variable y deberán hacer referencia al contenido de la página.

```
<meta name="keywords" content="editorial, marcas, html"/>
```

- **«author»**. Permite indicar el nombre del autor.

```
<meta name="author" content="El nombre del autor" />
```

- **«copyright»**. Permite indicar el propietario de los derechos de la página web.

```
<meta name="copyright" content="Ediciones Paraninfo" />
```

© Ediciones Paraninfo

- **«viewport».** Permite determinar el ancho de la zona de representación *(viewport)* de la página, la escala inicial o las posibilidades de escalado que se le aplican a la página.

```
<meta name="viewport" content="width=device-width, initial-scale=1.0">
```

Nota técnica

URL son las siglas de *Uniform Resource Locator* («localizador de recursos uniforme»). Consiste en una secuencia de caracteres que identifica un recurso dentro de una red, normalmente internet, y permite su localización. Una URL puede almacenar la dirección, por ejemplo, a una página HTML, una imagen, un vídeo o un documento XML.

Formato general de una URL:

`esquema://servidor/directorio/recurso`

Un ejemplo básico de URL:

`https://www.paraninfo.es/Contacto.aspx`

2.3.3. Elemento body

El elemento <body> delimita la sección que contiene los elementos que forman la parte visible de la página HTML. Es, por lo tanto, la sección más importante, ya que en ella se encuentran los contenidos que se van a presentar en el navegador.

El número de etiquetas que se pueden incluir en esta sección es ingente y en esta unidad se muestra una importante relación de estas.

Los elementos de la sección delimitada por el elemento <body> se clasifican en elementos **de bloque** y elementos **de línea.** Dichos tipos tienen las siguientes características.

- Elementos de bloque.
  - Ocupan todo el espacio de su elemento padre (su contenedor).
  - Salvo excepciones, permiten contener a otros elementos, ya sean estos de bloque o de línea.
  - Salvo excepciones, siempre comienzan y terminan con un salto de línea.
- Elementos de línea.
  - Ocupan lo que ocupa su contenido.
  - Contienen datos u otros elementos de línea.
  - Pueden comenzar en cualquier lugar de una línea y no generan salto de línea.

2.3.4. Elementos de bloque

Los elementos de bloque forman un conjunto muy heterogéneo, ya que tienen en común características relacionadas con el espacio que ocupan y su distribución en la página. No

© Ediciones Paraninfo

obstante, es una de las categorías que se pueden utilizar para clasificar los elementos de HTML.

En esta sección, se muestra la relación de elementos de bloque, ordenados alfabéticamente. Se incluye su descripción, referencias a los atributos más significativos (exceptuando los atributos globales), ocasionalmente código HTML de ejemplo junto con su representación visual y, cuando proceda, información complementaria que ayude a comprender el componente. La lista de atributos no es exhaustiva, ya que esta es muy extensa, por lo que se sugiere acudir a manuales de referencia o a documentación técnica para profundizar en las características de cada componente.

Nota técnica

MDN Web Docs es una web de la Fundación Mozilla que contiene recursos para desarrolladores. Se puede profundizar en el conocimiento de HTML en general y de los elementos de bloque en particular a través de sus páginas.

Información relativa a los elementos en bloque:

https://developer.mozilla.org/es/docs/Web/HTML/Block-level_elements

Los elementos de bloque permiten, entre otras cosas, estructurar y organizar la página para facilitar su correcta interpretación por parte de los programas informáticos. Una posible estructura de página se muestra en la Figura 2.4. En dicha estructura, el contenido del bloque HEADER podría contener, por ejemplo, los enlaces para la navegación general; el bloque ASIDE, los enlaces para navegación referente a la sección en la que se encuentre el usuario; el bloque SECTION, una sección contenedora de dos artículos incluidos en sendos bloques ARTICLE, y el bloque FOOTER, podría incluir referencias a la empresa propietaria de los derechos de la página, además de otra información.

Cada uno de los contenedores, desde el más general hasta el más específico, es un elemento de bloque.

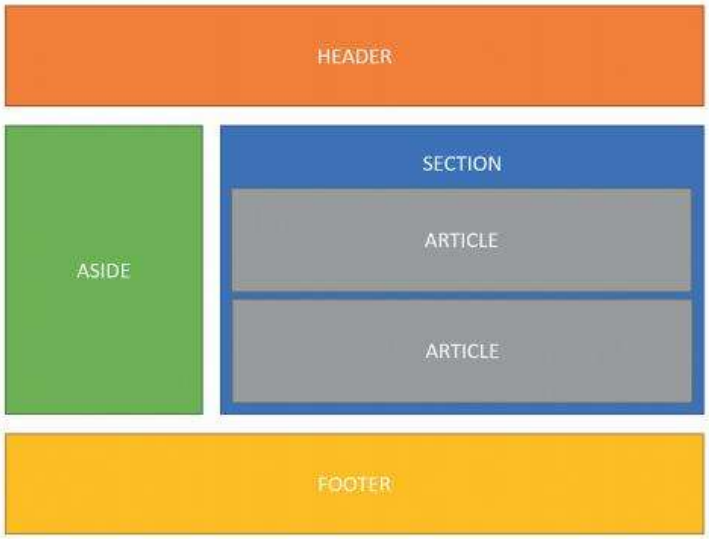


Figura 2.4. *La estructura de una página HTML la determina su autor.*

© Ediciones Paraninfo

Esta estructura, como se ha comentado, es una referencia y admite múltiples variantes. Lo realmente importante es ubicar cada contenido en un bloque semántico correspondiente para facilitar el procesamiento automático del documento.

Recuerda

Muchos de los elementos bloques no tienen una representación gráfica característica, siendo iguales entre sí. La importancia de estos elementos se encuentra en la semántica que proporcionan, ya que dan indicaciones de su contenido para poder realizar un procesamiento automático más sencillo y eficaz.

Se detallan a continuación los elementos que se engloban en esta categoría, excepto los elementos **<form>** y **<fieldset>** que se exponen en el Apartado 2.5 y el elemento **<table>**, que se expone en el Apartado 2.4.

<address>

Descripción: representa información de contacto (dirección de correo electrónico, número de teléfono, etc.) para el elemento **<article>** más cercano o para el elemento **<body>** en el que se encuentra. El texto se muestra en cursiva.

Ejemplo:

```
<address>
Puede contactar con la editorial a través del sitio web <a href="https://www.
paraninfo.es/">www.paraninfo.es</a>
</address>
```

Puede contactar con la editorial a través del sitio web www.paraninfo.es

Figura 2.5. *El elemento <address> indica que su contenido es información de contacto.*

<article>

Descripción: este contenedor agrupa información concebida como una unidad independiente. Algunos ejemplos serían una noticia de un periódico, un artículo de una revista, una entrada en un blog o un comentario en un foro. No genera ninguna representación gráfica específica.

<aside>

Descripción: agrupa contenidos que están relacionados con el contenido principal del documento. Al ser un elemento de orientación puramente semántica, no tiene representación visual específica. No obstante, se representa habitualmente como una barra lateral

© Ediciones Paraninfo

en la que se muestran enlaces relacionados, publicidad o información complementaria al contenido principal del documento.

<blockquote>

Descripción: contiene una cita.

Atributos significativos:

- **cite:** contiene la referencia a la fuente de la cita.

Ejemplo:

```
<blockquote cite='https://webfoundation.org/2017/03/web-turns-28-letter/'>
    <p>El modelo comercial actual de muchos sitios web ofrece contenido
gratuito a cambio de datos personales. Muchos de nosotros estamos de acuerdo
con esto, aunque a menudo al aceptar documentos de términos y condiciones
largos y confusos, pero fundamentalmente no nos importa que se recopile
cierta información a cambio de servicios gratuitos.</p>
</blockquote>
```

<details>

Descripción: almacena un contenido textual que se hace visible cuando se «despliega» el componente.

Puede incluir como hijo el elemento **<summary>** para representar una etiqueta que identifique el contenido. De no usarse **<summary>**, se mostrará una etiqueta genérica.

Se trata de una tecnología experimental, por lo que algunos navegadores pueden no interpretar este elemento correctamente.

Ejemplo:

```
<details>
        <summary>Nota para el usuario</summary>
     El contenido de este manual está actualizado en relación a la versión
10.8 del producto.
</details>
```

▶ Nota para el usuario

Figura 2.6. *El elemento <details> permite ocultar el contenido para que no ocupe espacio.*

▼ Nota para el usuario
El contenido de este manual está actualizado en relación a la versión 10.8 del producto.

Figura 2.7. *Al interactuar con el elemento <details> se muestra el contenido oculto.*

© Ediciones Paraninfo

<dialog>

Descripción: muestra una caja de diálogo.

Atributos significativos:

- open: si está disponible, se muestra el diálogo; en caso contrario, no se muestra.

Ejemplo:

```
<dialog open>
        <p>¡La compra se ha realizado correctamente!</p>
</dialog>
```

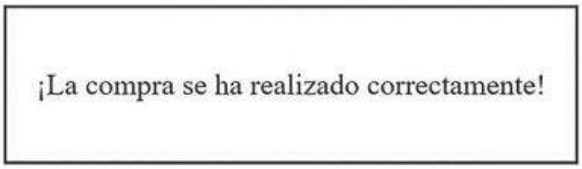


Figura 2.8. *Las cajas de diálogo son útiles para captar la atención del usuario.*

Descripción: permite crear divisiones y agrupar contenidos, sean estos del tipo que sean. Se utiliza, junto con las hojas de estilo CSS, para maquetar y estructurar las páginas. No tiene una representación visual particular.

<dl>, <dt> y <dd>

Descripción: usados conjuntamente permiten definir una lista de descripciones. **<dl>** marca la lista de definición, **<dt>** cada uno de los términos definidos y **<dd>** la definición de cada término.

Ejemplo:

```
<p>Géneros de videojuegos:</p>
<dl>
     <dt>Arcade</dt>
     <dd>Tienen como objetivo alcanzar una meta salvando obstáculo y/o
enemigos. Suelen ser muy dinámicos y tienen mecánicas sencillas.</dd>
     <dt>Estrategia</dt>
     <dd>Tienen como objetivo organizar una serie de recursos y maximizar su
rendimiento, ya sea este bélico, económico, social, etc.</dd>
     <dt>Disparos</dt>
     <dd>Tiene como objetivo alcanzar una meta o cumplir una misión en un
entorno hostil. También conocidos como <em>shooters</em></dd>
</dl>
```


© Ediciones Paraninfo

Géneros de videojuegos:

Arcade
Tienen como objetivo alcanzar una meta salvando obstáculos y/o enemigos. Suelen ser muy dinámicos y tienen mecánicas sencillas.
Estrategia
Tienen como objetivo organizar una serie de recursos y maximizar su rendimiento, ya sea este bélico, económico, social, etc.
Disparos
Tiene como objetivo alcanzar una meta o cumplir una misión en un entorno hostil. También conocidos como *shooters.*>

Figura 2.9. *La lista de definición agrupa conceptos y descripciones.*

Actividad propuesta 2.4

Creación de una lista de descripciones

Utilizando los elementos <dl>, <dt> y <dd>, crea un documento HTML que contenga una lista con las descripciones de los siguientes términos: internet, WWW y HTML.

<figure> y <figcaption>

Descripción: representan una ilustración, figura, diagrama, etc., y su leyenda o subtítulo (mediante el elemento **<figcaption>**).

Ejemplo:

```
<figure>
        <img src="logow3c.png" alt="Logotipo del W3C">
        <figcaption>El logotipo del World Wide Web Consortium (W3C)</
figcaption>
</figure>
```

El logotipo del World Wide Web Consortium (W3C)

Figura 2.10. *Utilizando el elemento <figcaption> se mejora la accesibilidad de la página web.*

Nota técnica

La accesibilidad de una página web es la capacidad que tiene para garantizar el acceso a la información contenida en ella a personas con algún tipo de discapacidad. El W3C define una serie de pautas para garantizar que una página es accesible. Puedes profundizar en la accesibilidad web a través de este enlace: https://www.w3c.es/Traducciones/es/WAI/intro/accessibility

© Ediciones Paraninfo

<footer>

Descripción: este contenedor representa un pie de página. Típicamente contiene información sobre la propiedad, autoría y los derechos del documento. Es un elemento semántico por lo que agrupa, pero no proporciona ninguna aportación visual.

<h1>, <h2>, <h3>, <h4>, <h5>, <h6>

Descripción: representan elementos de encabezado. Tienen distintos niveles de importancia. El elemento **<h1>** es el más importante y el elemento **<h6>**, el menos importante.

Cada elemento tiene una representación distinta en el navegador (a mayor importancia, mayor tamaño del texto) que se puede replicar mediante hojas de estilo, pero esto no es una buena práctica.

Los agentes de usuario utilizan los elementos de encabezado para construir las tablas de contenido, lo que mejora la visibilidad de la página en los buscadores como Google. Utilizar estos elementos mejora, además, el posicionamiento (SEO), por lo que es muy importante utilizarlos de manera correcta.

Ejemplo:

```
<h1>Encabezado de nivel 1</h1>
<h2>Encabezado de nivel 2</h2>
<h3>Encabezado de nivel 3</h3>
<h4>Encabezado de nivel 4</h4>
<h5>Encabezado de nivel 5</h5>
<h6>Encabezado de nivel 6</h6>
```

Encabezado de nivel 1

Encabezado de nivel 2

Encabezado de nivel 3

Encabezado de nivel 4

Encabezado de nivel 5

Encabezado de nivel 6

Figura 2.11. *Los elementos de encabezado son muy importantes para obtener un buen posicionamiento.*

© Ediciones Paraninfo

Nota técnica

Generalmente se conoce como **agente de usuario** a la aplicación que está accediendo como cliente a un recurso proporcionado por un servidor web. Los navegadores web son ejemplos de agentes de usuario, pero también lo son los robots de búsqueda o de compra. En general, cualquier programa que acceda a contenido alojado en un servidor web de forma autónoma o dirigida por un usuario se puede considerar agente de usuario.

<header>

Descripción: contiene el encabezado de un documento o de una sección. En este elemento, de naturaleza semántica, se suelen ubicar los logotipos e imágenes corporativas, los formularios de búsqueda, la cesta de la compra en sitios web de venta online o enlaces de navegación de alto nivel. No tiene una representación visual específica.

<hr>

Descripción: representa un cambio de tema entre párrafos mediante una línea de división. Aunque se representa como una línea, su principal objetivo es aportar información semántica. No puede ser contenedor.

Ejemplo:

```
<p>Este texto está en un párrafo.</p>
<hr>
<p>Este texto está en otro párrafo.</p>
```

Este texto está en un párrafo.

Este texto está en otro párrafo.

Figura 2.12. *Es importante que la separación entre bloques de información sea visible.*

<li>, <ol>, <ul>

Descripción: representan una lista y sus elementos. La etiqueta **<li>** identifica cada uno de los elementos de la lista. Por su parte, las etiquetas **<ol>** y **<ul>** identifican una lista ordenada (sus elementos aparecen numerados) y una lista no ordenada (sus elementos aparecen identificados con viñetas). Las listas se pueden anidar.

© Ediciones Paraninfo

Ejemplo (lista ordenada):

```
<ol>
    <li>Subir al Kilimanjaro</li>
    <li>Visitar Nueva York</li>
    <li>Aprender a jugar al ajedrez</li>
    <li>Viajar en el tren Transiberiano </li>
</ol>
```

1. Subir al Kilimanjaro
2. Visitar Nueva York
3. Aprender a jugar al ajedrez
4. Viajar en el tren Transiberiano

Figura 2.13. *Las listas ordenadas se utilizan en relaciones de elementos cuyo orden es importante.*

Ejemplo (lista no ordenada):

```
<ul>
    <li>Lunes</li>
    <li>Martes</li>
    <li>Miércoles</li>
    <li>Jueves</li>
    <li>Viernes</li>
</ul>
```

- Lunes
- Martes
- Miércoles
- Jueves
- Viernes

Figura 2.14. *Las listas sin ordenar se utilizan en relaciones de elementos cuyo orden no es importante.*

Actividad propuesta 2.5

Creación de una lista numerada

Crea un documento HTML que contenga una lista numerada en la que se muestren los pasos que se han de seguir para cocinar una tortilla de patatas.

<main>

Descripción: representa el contenido principal del documento. La motivación de esta etiqueta es principalmente semántica, ya que identifica de manera inequívoca el bloque de

© Ediciones Paraninfo

información principal de la página. Debe incluirse un único contenedor **main** dentro de un documento, y este no debe estar dentro de elementos **<article>**, **<aside>**, **<footer>**, **<header>** ni **<nav>**, por razones semánticas. No tiene representación específica.

<nav>

Descripción: este contenedor está concebido para albergar enlaces de navegación, ya sean estos internos al documento o a otros documentos. Habitualmente se muestran las tablas de contenido, los índices, menús o sistemas de navegación alternativos. No tiene representación específica.

<p>

Descripción: delimita un párrafo de texto. No tiene representación específica.

<pre>

Descripción: representa un texto tal y como está en el documento HTML, respetando saltos de línea, tabulaciones y espacios múltiples. Utiliza una fuente de letra de ancho fijo.

Ejemplo:

```
<p>
    Este texto está dentro de un párrafo.
    Las tabulaciones, saltos de línea y los espacios       múltiples se ignoran.
</p>
<pre>
Este texto está dentro de un elemento pre.
        Los saltos de línea y los espacios       múltiples se respetan.
</pre>
```

Este texto está dentro de un párrafo. Las tabulaciones, saltos de línea y los espacios múltiples se ignoran.

```
Este texto está dentro de un elemento pre.
        Los saltos de línea y los espacios       múltiples se respetan.
```

Figura 2.15. *Solo se deben respetar espacios, tabulaciones y saltos de línea múltiple cuando sea necesario.*

<section>

Descripción: representa una sección genérica del documento. En ocasiones su uso se confunde con el de **<article>**, pero su diferencia es importante. Los elementos agrupados por **<article>** están relacionados entre sí y forman un bloque autocontenido e independiente de información. En cambio, **<section>** tiene como objetivo agrupar información relacionada y dependiente, aportando estructura semántica. No tiene una representación específica.

© Ediciones Paraninfo

2.3.5. Elementos de línea

Los elementos en línea se denominan así porque se integran entre los elementos anterior y posterior sin introducir saltos de línea, generando continuidad y ocupando solo el espacio necesario para mostrar su contenido.

Enlaces, abreviaturas o marcas son algunos ejemplos de elementos en línea con comportamientos, significados o estilos concretos. Otros elementos en línea, como , son simples delimitadores que posibilitan la aplicación de estilos mediante CSS sin necesitar añadir otros elementos de forma innecesaria.

Se detallan a continuación los elementos que se engloban en esta categoría, excepto los elementos **<audio>**, **<canvas>**, **<img>**, **<map>**, **<svg>** y **<video>**, que se exponen en el Apartado 2.6 dedicado a los elementos multimedia; **<button>**, **<datalist>**, **<embed>**, **<input>**, **<label>**, **<meter>**, **<output>**, **<progress>**, **<select>** y **<textarea>**, que se exponen en el Apartado 2.5 dedicado a los formularios y **<iframe>**, **<object>** y **<picture>**, que se exponen en la sección dedicada a los contenidos incrustados, dentro de esta misma unidad.

<a>

Descripción: representa un enlace *(anchor)* a una URL de internet, ya sea una página, un correo electrónico o cualquier otro recurso. Al pulsar sobre el contenido marcado con esta etiqueta, el navegador se dirigirá a la URL indicada o realizará la acción correspondiente si no se trata de una dirección de navegación (por ejemplo, si el enlace es a una dirección de correo electrónico, abrirá el programa de gestión de correos instalado en el ordenador).

Salvo que se indique lo contrario, y siempre que tenga el atributo **href**, el texto del enlace aparecerá subrayado y en un color predeterminado por el estilo asignado por defecto o el asignado mediante estilos CSS. Además, al pasar el ratón por encima, el aspecto del cursor cambiará para indicar que se trata de un enlace.

Los enlaces se pueden clasificar en función de la dirección a la que están referenciando:

- **Externos.** Referencian a un recurso ubicado en un servidor distinto de aquel en el que está alojada la página HTML que contiene el enlace.
- **Internos.** Referencia a un recurso ubicado en el mismo servidor en el que está alojada la página HTML que contiene el enlace.
- **Internos y externos a elementos concretos.** Referencian al identificador de un elemento dentro del propio documento o de un documento distinto, ya sea este interno o externo. Utiliza el símbolo # para indicar el identificador del elemento al que se va a referenciar. La sintaxis es **<a href="nombre-pagina.html#identificador">**

Atributos significativos:

- **download:** si se añade este atributo, y en determinadas circunstancias (por ejemplo, que la URL haga referencia a una página HTML almacenada en el mismo servidor web), el navegador descargará el recurso al que apunta la URL en lugar de intentar abrirlo en el navegador. Dependiendo del navegador hará la descarga directamente o permitirá al usuario elegir abrir el recurso o descargarlo.

© Ediciones Paraninfo

- **href:** contiene la URL a la que apunta el enlace. Es imprescindible si se desea que el enlace exista y se muestre como tal (subrayado, coloreado y con el icono del ratón predeterminado). Puede ser una página HTML o cualquier otro recurso (una imagen, un documento en formato PDF, un fichero de audio, etc.). También puede hacer referencia a una dirección de correo electrónico (añadiendo el prefijo «mailto:» al valor del atributo) o a un número de teléfono (añadiendo el prefijo «tel:» al valor del atributo).
- **hreflang:** indica en qué idioma está escrita la página de destino del enlace.
- **ping:** permite indicar una o más URL sobre las que se realizarán peticiones de forma transparente cuando el usuario pulse sobre el enlace. Se suele utilizar para registrar la actividad del usuario.
- **rel:** determina la relación que existe entre el enlace y el recurso al que hace referencia.
- **target:** permite indicar dónde abrir la URL enlazada. Algunos valores típicos son: **_self**, que carga el recurso en el mismo contexto de navegación en el que está el enlace (típicamente la pestaña del navegador); **_blank**, que carga el recurso en un nuevo contexto de navegación (típicamente en una nueva pestaña del navegador); **_parent**, que carga el recurso en el contexto de navegación padre del actual, o **_top**, que carga el recurso en el contexto más alto de navegación.
- **type:** indica el tipo de contenido (conocido como tipo MIME, «MEDIA TYPE» o «CONTENT TYPE») del destino del enlace. Es un atributo informativo.

Ejemplo 1:

```
<p>
Una olla de algo más vaca que <a href="https://dle.rae.es/carnero" target="_
blank" type="text/html">carnero</a>, salpicón las más noches...
</p>
<p>
Escribir un email a <a href="mailto:cervantes@paraninfo.es">Miguel de
Cervantes</a>
</p>
```

Una olla de algo más vaca que carnero, salpicón las más noches...

Escribir un email a Miguel de Cervantes

Figura 2.16. *Los hiperenlaces o hipertextos son la piedra angular de la web, ya que permiten relacionar recursos y, por lo tanto, la navegación.*

Ejemplo 2:

```
<ul>
    <li><a href="#capitulo1">Sección 1</a></li>
    <li><a href="#capitulo2">Sección 2</a></li>
    <li><a href="#capitulo3">Sección 31</a></li>
```

© Ediciones Paraninfo

```
</ul>
<article id="capitulo1">
    <!-- CONTENIDO DEL CAPÍTULO 1-->
</article>
<article id="capitulo2">
    <!-- CONTENIDO DEL CAPÍTULO 2-->
</article>
<article id="capitulo3">
    <!-- CONTENIDO DEL CAPÍTULO 3-->
</article>
```

Nota técnica

La lista completa de tipos de medios (MEDIA-TYPES) se puede consultar en la siguiente página:

https://www.iana.org/assignments/media-types/media-types.xhtml

Actividad propuesta 2.6

Creación de enlaces

Crea un documento HTML que contenga enlaces a tres periódicos nacionales o extranjeros. Consigue que los enlaces se abran en pestañas nuevas.

<abbr>

Descripción: indica que el texto marcado es abreviatura o acrónimo. Se representa subrayado con una línea de puntos.

Atributos significativos:

- **title:** contiene la descripción completa de la abreviatura. Al poner el ratón sobre la abreviatura, se muestra el contenido de este atributo en un cuadro emergente *(tooltip)*.

Ejemplo:

```
De todos los videojuegos, el que más me gusta es el <abbr title="Call Of
Duty">COD</abbr>
```

De todos los videojuegos, el que más me gusta es el COD

Call Of Duty

Figura 2.17. *Es conveniente dar información sobre las abreviaturas al menos la primera vez que se utilizan.*

© Ediciones Paraninfo

Actividad propuesta 2.7

Abreviaturas

Crea un documento HTML que contenga un texto en el que se hable de HTML y la WWW. Utiliza el elemento <abbr> para incluir el significado de las dichas siglas.

<b>

Descripción: indica que el texto que contiene debe ser mostrado en **negrita.**

Ejemplo:

```
<p>
    Pulsa las teclas <b>Ctrl+C</b> para copiar al portapapeles.
</p>
```

Pulsa las teclas **Ctrl+C** para copiar al portapapeles.

Figura 2.18. *Con las negritas se capta la atención del usuario.*

<bdi> y <bdo>

Descripción: permiten establecer la dirección del texto (**<bdi>**) o anular el algoritmo que decide la dirección (**<bdo>**). Se suelen utilizar para escribir textos en lenguas que se escriben de derecha a izquierda (RTL), como el árabe o el hebreo, frente a la escritura realizada de izquierda a derecha (LTR), como el español o el inglés.

Nota técnica

RTL son las siglas de las palabras en inglés *Right To Left* («de derecha a izquierda»). Por su parte, LTR procede de *Left To Right* («de izquierda a derecha»).

Estas siglas se utilizan como valores para ciertos atributos. Por ejemplo, el atributo "direction" de CSS puede tomar los valores "ltr" y "rtl".

Descripción: produce un salto de línea. Los navegadores ignoran el salto de línea convencional utilizado en computación, representado con el carácter «\n», por lo que es necesario su uso para forzar a realizar un salto de línea. No se debe utilizar como elemento de separación de párrafos, ya que esto se debe realizar con el elemento **<p>**. Se debe utilizar cuando se desea hacer un salto entre líneas del mismo bloque de información (código fuente de un programa informático, un poema, una dirección postal, etcétera).

© Ediciones Paraninfo

Ejemplo:

```
<p>
    "Sí, repite la hormiga,<br>
    he visto las estrellas.<br>
    subí al árbol más alto<br>
    que tiene la alameda<br>
    y vi miles de ojos<br>
    dentro de mis tinieblas."<br>
</p>
<p>
    Extracto del poema 'Los encuentros de un caracol aventurero', de Federico
García Lorca.
</p>
```

"Sí, repite la hormiga,
he visto las estrellas.
subí al árbol más alto
que tiene la alameda
y vi miles de ojos
dentro de mis tinieblas."

Extracto del poema 'Los encuentros de un caracol aventurero', de Federico García Lorca.

Figura 2.19. *El elemento <br> crea un salto de línea manual, pero no desactiva los automáticos.*

<cite>

Descripción: indica una referencia a un autor o a una fuente de información. El texto se muestra en cursiva.

Ejemplo:

```
<p>Según el genial pintor Picasso <cite>"La inspiración existe, pero tiene
que encontrarte trabajando"</cite>.</p>
```

Según el genial pintor Picasso *"La inspiración existe, pero tiene que encontrarte trabajando"*.

Figura 2.20. *El navegador muestra las citas en letra cursiva.*

<code>

Descripción: indica que el texto contendido es código fuente de un programa informático, utilizando un tipo de letra de ancho fijo (monoespaciada).

© Ediciones Paraninfo

Ejemplo:

```
<p>
    En Java, con <code>Math.random()</code>, podemos generar un número
aleatorio.
</p>
```

En Java, con `Math.random()`, podemos generar un número aleatorio.

Figura 2.21. *Los tipos de letra de ancho fijo son los adecuados para los programas informáticos.*

Actividad propuesta 2.8

Código fuente

Crea un documento HTML que contenga un bloque de código escrito en el lenguaje de programación que estás aprendiendo. Utiliza los elementos <code> y <br> para conseguir que se muestre correctamente.

<data>

Descripción: vincula un contenido con una traducción comprensible por una máquina. No tiene efecto visible en la página. Su utilidad es facilitar su procesamiento automático.

Atributos significativos:

- **data:** contiene el valor asignado al elemento.

Ejemplo:

```
<ul>
    <li>Ingresos anuales: <data value="10500300">10500300€</data></li>
    <li>Número de unidades:<data value="5344">5000+</data> </li>
    <li>Vendedor del año: <data value="Calamaro, Diego">Diego Calamaro</
data></li>
</ul>
```

<del>

Descripción: permite marcar un contenido textual para indicar que ha sido borrado. El contenido se mostrará tachado, evidenciando que ha sido eliminado y, quizá, sustituido por otro más reciente o correcto. Se utiliza para permitir la trazabilidad en los cambios realizados sobre los documentos HTML.

Atributos significativos:

- **cite:** contiene la URL donde se explica la razón del cambio.
- **datetime:** contiene la fecha o la hora del cambio.

© Ediciones Paraninfo

Ejemplo:

```
<p>
El lenguaje de programación más utilizado es
<del datetime="2020-03-04 10:00:00"
cite="estudioLenguajes.html">Java</del> Python
</p>
```

El lenguaje de programación más utilizado es ~~Java~~ Python

Figura 2.22. *En ocasiones es conveniente mantener el estado original de la página web.*

<dfn>

Descripción: dentro de un elemento **<p>**, **<section>** o **<dl>** que definen un término, el elemento **<dfn>** identifica dicho término. Representa el término en letra cursiva para que se pueda identificar de manera visual.

Ejemplo:

```
<p>
    Los <dfn>videojuegos de plataformas</dfn> son muy dinámicos. El personaje
controlado por el jugador debe andar, correr, saltar, esquivar, deslizarse o
disparar para conseguir completar el objetivo del juego.
</p>
```

Los *videojuegos de plataformas* son muy dinámicos. El personaje controlado por el jugador debe andar, correr, saltar, esquivar, deslizarse o disparar para conseguir completar el objetivo del juego.

Figura 2.23. *El elemento <dfn> permite mejorar el posicionamiento de la página en los buscadores.*

Actividad propuesta 2.9

Definiciones

Crea un documento HTML que contenga las definiciones de los elementos <abbr> y <hr>. Utiliza los elementos <p> y <dfn>.

Descripción: permite enfatizar un texto. Visualmente, el navegador muestra dicho texto en cursiva. Pese a que la representación visual es la misma que la etiqueta **<i>**, la etiqueta

© Ediciones Paraninfo

**** tiene un significado semántico distinto, ya que indica que el autor está poniendo énfasis en la palabra o palabras marcadas.

Ejemplo:

```
<p>Te dije que la cita era <em>mañana</em>, no hoy.</p>
```

Te dije que la cita era *mañana*, no hoy.

Figura 2.24. *Los navegadores enfatizan un texto delimitado por la etiqueta mostrándolo en cursiva.*

<i>

Descripción: el texto delimitado por esta marca aparece en cursiva, ya que es diferente del resto por alguna razón. El resultado visual es el mismo que el generado por ****, pero sin su aportación semántica.

Ejemplo:

```
<p>En el Parque Nacional de Monfragüe anidan varios ejemplares de <i>Aegypius
monachus</i>, el ave de mayor envergadura de la península ibérica.</p>
```

En el Parque Nacional de Monfragüe anidan varios ejemplares de *Aegypius monachus*, el ave de mayor envergadura de la península ibérica.

Figura 2.25. *Aunque visualmente los elementos <i> y sean iguales, semánticamente son distintos.*

<ins>

Descripción: este elemento es el antagonista del elemento **<del>.** Indica que el contenido etiquetado ha sido añadido a una versión anterior del documento HTML. El texto marcado con este elemento aparece subrayado.

Atributos significativos:

- **cite:** contiene la URL donde se explica la razón del cambio.
- **datetime:** contiene la fecha o la hora del cambio.

Ejemplo:

```
<p>Los lenguajes de programación más demandados son Java, Python, C# y
<ins>Kotlin</ins>.</p>
```

© Ediciones Paraninfo

Los lenguajes de programación más demandados son Java, Python, C# y Kotlin.

Figura 2.26. *El elemento <ins> permite llamar la atención al usuario sobre el nuevo contenido.*

Actividad propuesta 2.10

Insertando y eliminando contenido

Crea un documento HTML que contenga una referencia a la versión actual de HTML. Consigue que aparezca la versión anterior tachada, para que quede claro que se ha actualizado la página. Utiliza los elementos <ins> y <del>.

<kbd>

Descripción: indica que el texto contenido entre las etiquetas de apertura y cierre debe introducirlo el usuario. La representación depende del navegador, pero siempre será distinta de la representación del texto por defecto.

Ejemplo:

```
Por favor, pulse <kbd>Ctrl+F1</kbd> para reiniciar el simulador
```

Por favor, pulse `Ctrl+F1` para reiniciar el simulador.

Figura 2.27. *Diferenciar visualmente algunos contenidos mejora la usabilidad.*

<mark>

Descripción: resalta un texto marcándolo en color amarillo, tal y como se hace con un rotulador sobre un texto en un papel.

Ejemplo:

```
<p>Rafael tiene alergia a las <mark>nueces</mark> y a los <mark>pistachos</
mark></p>
```

Rafael tiene alergia a las nueces y a los pistachos

Figura 2.28. *En ocasiones es necesario resaltar contenidos de manera evidente.*

<noscript>

Descripción: representa la alternativa a un bloque delimitado con el elemento **<script>.** Cuando el soporte para programas *(scripts)* no está disponible por cualquier razón (normal-

© Ediciones Paraninfo

mente por estar deshabilitado), se mostrará el contenido de este elemento. Si el soporte para programas está disponible, el contenido de este elemento será ignorado.

Actividad propuesta 2.11

Marcado

Crea un documento HTML que contenga una lista de los módulos profesionales que estás cursando durante este trimestre, utilizando los elementos <ul> y <li>. Resalta con el elemento <mark> los dos módulos que más te gustan.

Nota técnica

Informalmente, se conoce como *script* a un programa (que suele ser de pequeño tamaño) compuesto por una serie de comandos que ejecuta un intérprete.

En la programación web en entorno cliente, los *scripts* se escriben en JavaScript y se interpretan por el navegador.

<q>

Descripción: indica una cita literal. Los navegadores suelen entrecomillar el contenido textual delimitado por esta etiqueta.

Atributos significativos:

- **cite:** contiene la URL de la que se ha extraído la cita.

Ejemplo:

```
Como se indica en Wikipedia, <q cite="https://es.wikipedia.org/wiki/
HTML">HTML es un lenguaje de marcado que nos permite indicar la estructura de
nuestro documento mediante etiquetas</q>
```

Como se indica en Wikipedia, "HTML es un lenguaje de marcado que nos permite indicar la estructura de nuestro documento mediante etiquetas"

Figura 2.29. *Para preservar los derechos de los autores hay que identificar las citas.*

<s>

Descripción: muestra el texto delimitado por la etiqueta tachado con una línea horizontal, indicando que aquello a lo que hacía referencia ya no está vigente o disponible, o es incorrecto.

© Ediciones Paraninfo

Ejemplo:

```
<p><s>Nintendo Game Boy Color por 12.990 pesetas</s></p>
<p>Descatalogado</p>
```

~~Nintendo Game Boy Color por 12.990 pesetas~~

Descatalogado

Figura 2.30. *Si se mantienen contenidos obsoletos, hay que hacerlo saber.*

<samp>

Descripción: indica que el texto delimitado por la etiqueta es un mensaje proporcionado por un ordenador. Los navegadores, por defecto, utilizan una tipografía monoespaciada, por lo que visualmente el texto marcado con esta etiqueta se diferencia del texto de su entorno. Se suele utilizar de manera conjunta con el elemento **<pre>** para que se respeten espacios, tabuladores y saltos de línea.

Ejemplo:

```
<p>
Si introduce la dirección de manera incorrecta, el sistema mostrará la
siguiente información:
<pre><samp>
    Ha introducido una dirección incorrecta.
    Pulse F10 para reiniciar el proceso.
</samp></pre>
</p>
```

Si introduce la dirección de manera incorrecta, el sistema mostrará la siguiente información:

```
Ha introducido una dirección incorrecta.
Pulse F10 para reiniciar el proceso.
```

Figura 2.31. *Los textos correspondientes a mensajes generados por el ordenador se presentan en un tipo de letra diferente.*

Actividad propuesta 2.12

Manual de instrucciones

Crea un documento HTML que contenga un manual de instrucciones. En él se debe explicar que, para enviar un mensaje, hay que pulsar la tecla F2 y que el sistema responderá «Mensaje enviado satisfactoriamente» o «Ha ocurrido un error en el envío del mensaje». Utiliza el elemento <samp> para marcar los textos generados por el ordenador.

© Ediciones Paraninfo

<script>

Descripción: permite embeber código ejecutable en el documento HTML. Estos programas son interpretados y ejecutados por el navegador, por lo que no suponen una carga de trabajo para el servidor, ya que su entorno de ejecución es el dispositivo del cliente. Son potentes, muy versátiles y capaces de hacer prácticamente cualquier tarea. Estos programas se escriben utilizando JavaScript, pues es el lenguaje entendido por todos los navegadores web más utilizados.

Atributos significativos:

- **src:** contiene la URI del programa. Si existe, el código del programa no se encontrará en la página, sino que será cargado de un recurso externo.
- **type:** indica el lenguaje de programación utilizado. Si no se indica, el valor por defecto es JavaScript.

Ejemplo:

```
<script>
function sumar(a, b){
    var resultado = a + b;
    return resultado;
}
</script>
```

Nota técnica

URI es el acrónico de *Uniform Resource Identificator* («identificador de recurso universal»). Se trata de una cadena de caracteres que identifica de manera unívoca a un recurso dentro de una red, normalmente internet.

<slot>

Descripción: es una etiqueta con una motivación netamente programática, ya que permite definir componentes reutilizables.

<small>

Descripción: representa un comentario al margen, y reduce el tamaño del texto etiquetado. Se suele utilizar para mostrar avisos, atribuciones, derechos de propiedad, textos legales, etcétera.

Ejemplo:

```
Título: Cloud Computing, tecnología y negocio
<hr>
<small>Este contenido es propiedad de Ediciones Paraninfo</small>
```

© Ediciones Paraninfo

Título: Cloud Computing, tecnología y negocio

Este contenido es propiedad de Ediciones Paraninfo

Figura 2.32. *El texto etiquetado con <small> se diferencia por su menor tamaño.*

Descripción: el uso más habitual de este elemento consiste en delimitar una parte de un texto para asignarle un estilo distinto al del resto sin generar un nuevo bloque. Este elemento se utiliza cuando no existe otro elemento de línea más apropiado, por ejemplo **** o ****.

Ejemplo:

```
<p>
Una olla de algo más vaca que carnero, salpicón las más noches, duelos y
quebrantos los <span class="diassemana">sábados</span>, lantejas los <span
class="diassemana">viernes</span>, algún palomino de añadidura los <span
class="diassemana">domingos</span>, consumían las tres partes de su hacienda.
</p>
```

Una olla de algo más vaca que carnero, salpicón las más noches, duelos y quebrantos los **sábados**, lantejas los **viernes**, algún palomino de añadidura los **domingos**, consumían las tres partes de su hacienda.

Figura 2.33. *El elemento es una herramienta fundamental para maquetar páginas HTML.*

Nota técnica

Para que los textos delimitados por el elemento **** de la clase «**diassemana**» se muestren como se observa en la Figura 2.33, es necesario asignar un estilo a dicha clase. Aunque los estilos en cascada (CSS) se explicarán con detalle en la Unidad 3, se muestra a continuación el estilo aplicado en este ejemplo:

```
<style>
    .diassemana {
        font-weight: bolder;
        color: blue;
    }
</style>
```


Descripción: se utiliza para indicar importancia o severidad. Muestra el texto etiquetado en negrita con el objetivo de resaltarlo.

© Ediciones Paraninfo

Ejemplo:

```
<p>Para evitar una <strong>explosión</strong> es importante abrir las
ventanas</p>
```

Para evitar una **explosión** es importante abrir las ventanas

Figura 2.34. *El elemento permite poner énfasis en un bloque de texto.*

<sub> y <sup>

Descripción: permiten mostrar subíndices y superíndices.

Ejemplo:

```
<p>Para igualar la velocidad a x<sup>2</sup>, la aceleración debe ser
constante entre T<sub>0</sub> y T<sub>1</sub>.</p>
```

Para igualar la velocidad a x^2, la aceleración debe ser constante entre T_0 y T_1.

Figura 2.35. *Los índices y superíndices se utilizan a menudo en expresiones matemáticas.*

Actividad propuesta 2.13

Fórmula con índices

Crea un documento HTML que contenga la fórmula del físico Albert Einstein que relaciona la masa con la energía. Utiliza el elemento <sup>.

<template>

Descripción: permite crear fragmentos de documento (componentes) que pueden ser utilizados como plantillas. Es necesario utilizar JavaScript para rellenar estas plantillas, ya que ha de hacerse por programa, por lo que su explicación detallada excede el ámbito de este libro.

<time>

Descripción: el texto contenido en esta etiqueta representa un periodo de tiempo, ya sea una fecha, una hora o una duración. No tiene una representación visual específica; es una etiqueta semántica.

© Ediciones Paraninfo

Atributos significativos:

- **datetime:** contiene una representación del contenido destinada al procesamiento por parte de programas. Algunos de los valores que admite este atributo son los siguientes:
  - Año: 1993.
  - Mes y año: 1993-12.
  - Día, mes y año: 1993-12-10.
  - Año y semana de dicho año: 1993-W49.
  - Hora, minutos y segundos: 15:32:24.

Ejemplo:

```
<p>El Doom fue publicado el día <date datetime="1993-12-10">10 de diciembre
de 1993</date> y cambió para siempre el mundo de los videojuegos</p>
```

<var>

Descripción: indica que el texto marcado es el nombre de una variable de una expresión matemática o un programa informático. La variable aparece en letra cursiva.

Ejemplo:

```
<p>En este punto hay que modificar el valor de la variable <var>offset</var>
para desplazar la imagen del mago</p>
```

En este punto hay que modificar el valor de la variable *offset*
para desplazar la imagen del mago

Figura 2.36. *Las etiquetas con información semántica mejoran el posicionamiento y la accesibilidad.*

<wbr>

Descripción: indica al navegador que puede realizar un salto de línea si no hay espacio suficiente para representar la palabra completa (si hay espacio suficiente, este elemento no tiene efecto). No inserta el guion que convencionalmente indica que la palabra se ha «partido» para mostrarla en dos líneas. Si se desea que aparezca dicho guion, habrá que utilizar la entidad **** en lugar de **<wbr>.** Algunos navegadores pueden no tener soporte para este elemento.

Ejemplo:

En este ejemplo se muestran tres versiones del mismo código HTML. En la primera de ellas, no se indica el lugar adecuado para hacer el salto de línea. En la segunda versión, se indica el punto en el que se «sugiere» al navegador que haga un salto de línea. En la

© Ediciones Paraninfo

tercera, el salto de línea se indica mediante la entidad **** en lugar del elemento **<wbr>**, mostrando el navegador un guion en el punto en el que se ha realizado el salto de línea.

```
<p>Para obtener la referencia al elemento hay utilizar la sentencia
<code>document.getElementById("idElemento")</code> en el momento adecuado</p>
<p>Para obtener la referencia al elemento hay utilizar la sentencia
<code>document<wbr>.getElementById("idElemento")</code> en el momento
adecuado</p>
<p>Para obtener la referencia al elemento hay utilizar la sentencia
<code>document&shy;.getElementById("idElemento")</code> en el momento
adecuado</p>
```

Para obtener la referencia al elemento hay utilizar la sentencia `document.getElementById("idElemento")` en el momento adecuado

Para obtener la referencia al elemento hay utilizar la sentencia `document` `.getElementById("idElemento")` en el momento adecuado

Para obtener la referencia al elemento hay utilizar la sentencia `document-` `.getElementById("idElemento")` en el momento adecuado

Figura 2.37. *Habitualmente no es necesario modificar el comportamiento por defecto referente a los saltos de línea.*

2.3.6. Contenido incrustado

Un contenido incrustado es un elemento de una página HTML que está alojado en una ubicación diferente a la propia de la página, normalmente de otro servidor. Permite construir páginas complejas con elementos de otras fuentes, facilitando el diseño y el mantenimiento.

De manera resumida, un contenido incrustado es una referencia a un recurso almacenado en otra ubicación. Al tratarse de una referencia (en lugar de una copia), garantiza que si el contenido original se modifica, la actualización se realiza de manera automática e inmediata, ya que el origen de la información es el servidor en el que se encuentra alojada.

Se puede incrustar contenido como objetos multimedia, bloques de código HTML o contenidos diseñados por terceros mediante el uso del elemento <iframe>.

<embed>

Descripción: representa un punto de integración para un contenido externo incrustado (un *plug-in*) generado por otra aplicación. Con el objetivo de hacer accesible la página a más

© Ediciones Paraninfo

usuarios, no se recomienda el uso de contenidos externos incrustados desarrollados en tecnologías distintas de los estándares, ya que puede no estar soportado por todos los navegadores.

Atributos significativos:

- **src:** la URI del recurso que será incrustado y reproducido o ejecutado por el *plug-in*.
- **type:** tipo de contenido (también conocido como «tipo de medio de internet» o «tipo MIME»).
- **width** y **height:** determinan el tamaño del contenedor en el que se va a mostrar el contenido.

Ejemplo:

```
<embed type="video/webm" src="trailers/estreno.mov" width="640" height="480">
```

Figura 2.38. *Los vídeos son recursos multimedia muy útiles y fáciles de utilizar.*

Nota técnica

Se conoce coloquialmente como *plug-in* a los módulos o componentes que se añaden a un programa informático para ampliar su funcionalidad.

Existen muchos programas que disponen de una funcionalidad básica o esencial que, gracias a su diseño modular, admiten la incorporación de *plug-in* para dotarlos de nuevas funcionalidades. Los navegadores web son un claro ejemplo.

<iframe>

Descripción: representa un contexto de navegación anidado. Es una ventana que permite acceder a un recurso que se encuentra ubicado en un contexto distinto a la página que se está mostrando en el navegador. Básicamente, permite incrustar un documento o parte de un documento dentro de una página HTML.

© Ediciones Paraninfo

Atributos significativos:

- **src:** la URI del recurso incrustado.
- **srcdoc:** contiene el código HTML que mostrar dentro del elemento **<iframe>.** Si existe este atributo, se impone a la referencia proporcionada por el atributo **src.**
- **sandbox:** permite proporcionar restricciones al documento incrustado en el elemento **<iframe>** con objeto de mejorar la seguridad de la página.
- **allow:** directivas referentes a permisos de acceso a determinados elementos del dispositivo físico, tales como el acelerómetro, la cámara o la geolocalización.
- **allowfullscreen:** si existe, permite al contenido incrustado ocupar la pantalla completa.
- **width** y **height:** permiten indicar el ancho y el alto en píxeles.
- **loading:** con el valor **lazy** el contenido se carga cuando este se haga visible; con el valor **eager** (valor por defecto) el contenido se cargará inmediatamente.

Ejemplo 1:

En este primer ejemplo se incrusta un reproductor del portal de vídeos YouTube en un punto del documento HTML. El código del elemento **<iframe>** lo proporciona el propio portal YouTube, al igual que hacen muchos otros proveedores de contenidos como Vimeo, Google Maps o la Agencia Estatal de Meteorología, por citar solo algunos ejemplos.

```
<h3>Documental sobre fauna africana</h3>
<p>Pulsa en el centro del vídeo para comenzar la reproducción</p>
<iframe width="560" height="315" src="https://www.youtube.com/embed/
WNWVzg8ZfHg" frameborder="0" allow="accelerometer; autoplay; clipboard-write;
encrypted-media; gyroscope; picture-in-picture" allowfullscreen></iframe>
```

Documental sobre fauna africana

Pulsa en el centro del vídeo para comenzar la reproducción

Figura 2.39. *El componente <iframe> permite reutilizar contenido.*

© Ediciones Paraninfo

Ejemplo 2:

En este segundo ejemplo se incrusta un documento HTML completo dentro de otro documento HTML. En este caso, hay dos bloques de código. Un primer bloque consiste en la sección de la página que incrusta mediante un elemento **<iframe>** a otra página.

```
<p>Solución a la prueba práctica:</p>
<iframe width="300" height="100" src="solucion.html"></iframe>
```

Un segundo bloque corresponde a la página incrustada. Como se puede observar, es una página normal.

```
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Tabla de ingresos</title>
</head>
<body>
    <table>
        <tr>
            <th>Semestre</th>
            <th>Ingresos (€)</th>
        </tr>
        <tr>
            <td>Primero</td>
            <td>100.000</td>
        </tr>
        <tr>
            <td>Segundo</td>
            <td>101.700</td>
        </tr>
    </table>
</body>
</html>
```

Solución a la prueba práctica:

Semestre	Ingresos (€)
Primero	100.000
Segundo	101.700

Figura 2.40. *Con <iframe> se puede simplificar el mantenimiento de las páginas HTML.*

© Ediciones Paraninfo

Actividad propuesta 2.14

Contenido incrustado

Crea un documento HTML que contenga un vídeo musical obtenido de YouTube. Para incrustarlo en el documento, utiliza el código del elemento <iframe> que proporciona el propio portal de vídeos.

Para obtener el código del <iframe> de YouTube, accede desde el navegador, encuentra el vídeo y cuando se esté reproduciendo, selecciona la acción «Compartir» y después la opción «Insertar». Aparecerá entonces el código del elemento <iframe>, que deberás copiar y pegar en el cuerpo del documento HTML.

<object>

Descripción: permite incrustar un recurso externo que puede ser una imagen (una alternativa al componente **<img>**), un contexto de navegación anidado (una alternativa al componente **<iframe>**) o un recurso manejado por un *plug-in* (una alternativa al componente **<embed>**). El elemento **<param>** permite parametrizar el contenido incrustado.

Es importante tener en cuenta que no todos los navegadores se comportan igual en relación con los contenidos incrustados. Si no pueden reproducirlos por no disponer del complemento adecuado o por cuestiones de seguridad, podrán avisar mediante un mensaje o ignorar el componente. Dicho mensaje puede personalizarse si se agrega contenido HTML entre las etiquetas de apertura y cierre.

Atributos significativos:

- **data:** contiene el recurso que se ha de mostrar.
- **type:** permite indicar el tipo de contenido (tipo MIME) que se va a mostrar.
- **name:** es un nombre que podrá ser usado por elementos del contexto de navegación incrustado.
- **usemap:** es una referencia al atributo **name** de un elemento **<map>** para establecer una relación entre ambos.
- **form:** es una referencial al atributo **id** del formulario relacionado con el elemento.
- **width** y **height:** determinan, respectivamente, el ancho y alto del objeto incrustado.

Ejemplo 1:

En este ejemplo se incrustan cuatro componentes **<object>** que contienen una imagen, una página HTML, un vídeo y un documento en formato PDF.

```
<p>
<object data="seta.jpg" width="300" height="200"></object>
<object data="solucion.html" width="200" height="200"></object>
</p>
<p>
```

© Ediciones Paraninfo

```
<object data="trailer.mp4" width="300" height="200"></object>
<object data="BOE-A-2015-3178-consolidado.pdf" width="300" height="200"></
object>
</p>
```

Figura 2.41. *El elemento <object> admite múltiples alternativas.*

Ejemplo 2:

En este ejemplo se intenta incrustar un contenido que no es compatible con el navegador.

```
<object data="BeethovenElisa.mid" type="audio/midi"></object>
```

Figura 2.42. *No todos los navegadores interpretan los diferentes tipos de <object> de igual modo.*

© Ediciones Paraninfo

Ejemplo 3:

En este ejemplo se intenta incrustar un contenido que no es compatible con el navegador, pero a diferencia del ejemplo anterior, se proporciona un contenido alternativo para mostrarlo si el navegador no soporta el contenido proporcionado.

```
<object data="BeethovenElisa.mid" type="audio/midi"><em>¡Atención!</em> Este
contenido no es compatible con el navegador.</object>
```

¡Atención! Este contenido no es compatible con el navegador.

Figura 2.43. *Es recomendable proporcionar contenido alternativo en caso de incompatibilidad.*

<param>

Descripción: permite especificar parámetros de presentación de un elemento **<object>.**

Atributos significativos:

- **name:** contiene el nombre del parámetro.
- **value:** contiene el valor del parámetro.

<picture>

Descripción: utilizada de manera conjunta con el componente **<source>**, permite proporcionar versiones alternativas a un elemento **<img>** en función de una regla definida por el diseñador de la página. Es un componente que facilita la creación de imágenes adaptativas mejorando así la presentación de la información en función del tamaño del dispositivo, ya sea este un ordenador personal, una tableta, un teléfono inteligente *(smartphone)* o de cualquier otro tipo.

Ejemplo:

En este ejemplo se muestran tres imágenes de distinto tamaño, encuadre e, incluso, formato, en función del tamaño del navegador. Por defecto, se muestra el contenido proporcionado por el atributo **src** del elemento **<img>.** Las otras dos imágenes se muestran cuando el ancho máximo del navegador sea de 600 píxeles o de 1000 píxeles, respectivamente.

```
<article>
<p>
    En la imagen que ilustra esta notica se pueden observar las consecuencias
de la tormenta del pasado viernes.
</p>
<picture>
```

© Ediciones Paraninfo

```
    <source srcset="tormentaramas.jpg" media="(max-width: 600px)">
    <source srcset="tormentadetalle.png" media="(max-width: 1000px)">
    <img src="tormenta.jpg" alt="Arbol arrancado por la tormenta">
</picture>
</article>
```

En la imagen que ilustra esta noticia se pueden observar las consecuencias de la tormenta del pasado viernes.

Figura 2.44. *Esta imagen se muestra para un ancho máximo del navegador de 600 píxeles.*

En la imagen que ilustra esta noticia se pueden observar las consecuencias de la tormenta del pasado viernes.

Figura 2.45. *Esta imagen se muestra para un ancho máximo del navegador de 1000 píxeles.*

© Ediciones Paraninfo

En la imagen que ilustra esta noticia se pueden observar las consecuencias de la tormenta del pasado viernes.

Figura 2.46. *Esta imagen se muestra para un ancho del navegador a partir de 1000 píxeles.*

<source>

Descripción: permite proporcionar versiones alternativas a un elemento multimedia de tipo **<picture>**, **<audio>** o **<video>**. Debido a que no todos los navegadores admiten los mismos tipos de formato de audio y vídeo, en ocasiones el uso del elemento **<source>** es necesario para optimizar la compatibilidad.

Atributos significativos:

- **media:** solo válido para elementos **<picture>**. Permite determinar las reglas (utilizando *media queries*) para seleccionar una u otra imagen.
- **srcset** y **sizes:** solo válido para elementos **<picture>**. Permiten asignar varias imágenes junto con indicaciones para ayudar al navegador a elegir la versión más adecuada.
- **src:** indica la ubicación del recurso de audio o vídeo proporcionado por el componente.
- **type:** el tipo de contenido al que hace referencia el elemento.

Ejemplo:

Este código proporciona dos alternativas al mismo fichero de sonido.

```
<audio controls>
    <source src="/audio/caminandosobrelashojas.ogg" type="audio/ogg">
    <source src="/audio/caminandosobrelashojas.mp3" type="audio/mpeg">
</audio>
```

© Ediciones Paraninfo

Actividad resuelta 2.2

Creación de la página principal de un periódico digital

Crear la página principal de un periódico digital.

- El periódico estará compuesto por varias noticias.
- Las noticias estarán formadas por el título y el texto.
- Habrá una noticia de máxima relevancia y el resto serán de menor relevancia.

Solución

```
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <title>Document</title>
</head>
<body>
    <article>
        <h1>Título de la noticia principal</h1>
        <p>Texto de la noticia principal</p>
    </article>
    <article>
        <h2>Título de la primera noticia secundaria</h2>
        <p>Texto de la primera noticia secundaria</p>
    </article>
    <article>
        <h2>Título de la segunda  noticia secundaria</h2>
        <p>Texto de la segunda noticia secundaria</p>
    </article>
</body>
</html>
```

2.4. Tablas en HTML: estructura y elementos

Las tablas son elementos de diseño muy útiles para mostrar información organizada en filas y columnas. En el pasado las tablas se utilizaban, además, para realizar la maquetación general de las páginas, dada la sencillez que proporcionan para organizar los contenidos, pero es una práctica absolutamente desaconsejada.

2.4.1. Elementos de las tablas

Las tablas en HTML se forman con elementos específicos que no se deben utilizar fuera de su contexto. Estos elementos permiten construir estructuras visuales basadas en filas y columnas para la representación de datos. La flexibilidad que proporcionan los diferentes elementos es prácticamente absoluta, permitiendo construir tablas con distintas características.

© Ediciones Paraninfo

<table>

Descripción: este elemento delimita una tabla. Es un simple contenedor, ya que todo el contenido de la tabla está formado por elementos.

Ejemplo:

```
<table>
    <caption>Gastos anuales</caption>
    <thead>
        <tr>
            <th>Concepto</th>
            <th>Primer semestre</th>
            <th>Segundo semestre</th>
            <th>Total</th>
        </tr>
    </thead>
    <tfoot>
        <tr>
            <th>Total</th>
            <th>2650</th>
            <th>2080</th>
            <th>4730</th>
        </tr>
    </tfoot>
    <tbody>
        <tr>
            <td>Alimentación</td>
            <td>1500</td>
            <td>1750</td>
            <td>3250</td>
        </tr>
        <tr>
            <td>Tecnología</td>
            <td>850</td>
            <td>30</td>
            <td>880</td>
        </tr>
        <tr>
            <td>Deporte</td>
            <td>300</td>
            <td>300</td>
            <td>600</td>
        </tr>
    </tbody>
</table>
```

© Ediciones Paraninfo

Gastos anuales

Concepto	Primer semestre	Segundo semestre	Total
Alimentación	1500	1750	3250
Tecnología	850	30	880
Deporte	300	300	600
Total	**2650**	**2080**	**4730**

Figura 2.47. *Las tablas son herramientas fundamentales en la construcción de páginas HTML.*

<caption>

Descripción: proporciona un título descriptivo a una tabla. Debe ser declarado una única vez al comienzo de la tabla. El título, por defecto, se muestra en la parte superior de la tabla.

Ejemplo:

```
<caption>Gastos anuales</caption>
```

<thead>

Descripción: contiene el conjunto de filas que forman el encabezado de una tabla. Solo debe incluirse una vez por tabla y tiene que ubicarse inmediatamente antes del cuerpo (elemento **<tbody>**).

Ejemplo:

```
<thead>
    <tr>
        <th>Concepto</th>
        <th>Primer semestre</th>
        <th>Segundo semestre</th>
        <th>Total</th>
    </tr>
</thead>
```

<th>

Descripción: delimita una celda del encabezado de la tabla. Los elementos **<th>** suelen ubicarse en elementos **<tr>** que se encuentran, a su vez, en la sección de encabezado **<thead>** o de pie **<tfoot>.** Los textos marcados con esta etiqueta se muestran en negrita.

<tbody>

Descripción: delimita el cuerpo de una tabla (las filas que contienen de datos). Una tabla puede contener varios elementos **<tbody>** que delimiten diferentes grupos de datos.

© Ediciones Paraninfo

<colgroup>

Descripción: permite crear grupos de una o más columnas. También permite asignar atributos comunes a todas las columnas agrupadas. La agrupación se realiza de izquierda a derecha de manera consecutiva utilizando el atributo **span** o agregando elementos **<col>**.

Atributos significativos:

- **span:** determina el número de columnas consecutivas agrupadas por el elemento **<colgroup>**. Este número debe ser un entero positivo.

Ejemplo:

Sobre la declaración de la tabla del ejemplo anterior, incluyendo el siguiente código, se puede observar el funcionamiento de **<colgroup>** al asignar un estilo.

```
<table>
    <colgroup span="2" style="background-color: bisque;"></colgroup>
    <colgroup span="2" style="background-color:greenyellow;"></colgroup>
    <caption>Gastos anuales</caption>
```

Gastos anuales

Concepto	Primer semestre	Segundo semestre	Total
Alimentación	1500	1750	3250
Tecnología	850	30	880
Deporte	300	300	600
Total	**2650**	**2080**	**4730**

Figura 2.48. *El uso de estilos CSS permite modificar el aspecto de las tablas.*

<col>

Descripción: permite indicar algunas propiedades a las columnas o grupos de columnas de una tabla, tales como el color de fondo o el ancho. Los elementos **<col>** deben estar contenidos dentro de un elemento **<colgroup>**.

```
<table>
    <colgroup>
        <col style="width: 10em;">
        <col span="2" style="background-color: khaki;">
    </colgroup>
    <caption>Gastos anuales</caption>
```

© Ediciones Paraninfo

	Gastos anuales		
Concepto	**Primer semestre**	**Segundo semestre**	**Total**
Alimentación	1500	1750	3250
Tecnología	850	30	880
Deporte	300	300	600
Total	**2650**	**2080**	**4730**

Figura 2.49. *La flexibilidad del componente <table> es absoluta.*

<tr>

Descripción: representa una fila de una tabla. Es el elemento contenedor de las celdas de la tabla, ya sean estas de encabezado (**<th>**) o de cuerpo (**<td>**).

Ejemplo:

```
<tr>
    <td>Deporte</td>
    <td>300</td>
    <td>300</td>
    <td>600</td>
</tr>
```

<td>

Descripción: representa una celda del cuerpo de la tabla.

Atributos significativos:

- **colspan:** el número de celdas consecutivas hacia la derecha que debe agrupar.
- **rowspan:** el número de celdas consecutivas hacia abajo que debe agrupar.

Ejemplo:

```
<table>
    <tr>
        <td>Columna 1</td>
        <td>Columna 2</td>
        <td>Columna 3</td>
        <td>Columna 4</td>
    </tr>
    <tr>
```


© Ediciones Paraninfo

```
        <td colspan="2">F1 C1, C2</td>
        <td rowspan="2">F1, F2 C3</td>
        <td>F1 C4</td>
    </tr>
    <tr>
        <td>F2 C1</td>
        <td>F2 C2</td>
        <td>F2 C4</td>
    </tr>
</table>
```

Columna 1	Columna 2	Columna 3	Columna 4
F1 C1, C2		F1, F2 C3	F1 C4
F2 C1	F2 C2		F2 C4

Figura 2.50. *Los elementos <colspan> y <rowspan> permiten modificar la estructura de la tabla.*

<tfoot>

Descripción: delimita el conjunto de filas que forman el pie de una tabla. En esta sección normalmente se muestran los acumulados de las columnas o resúmenes de los datos mostrados en el elemento **<body>**.

```
<tfoot>
    <tr>
        <th>Total</th>
        <th>2650</th>
        <th>2080</th>
        <th>4730</th>
    </tr>
</tfoot>
```

Actividad resuelta 2.3

Creación de una tabla con datos sobre consolas de juegos

Crear una página HTML que contenga una tabla que muestre, para las consolas PS1, PS2, PS3, PS4 y PS5, el año de lanzamiento y el año de retirada.

Solución

```
<!DOCTYPE html>
<html lang="es">
<head>
```

© Ediciones Paraninfo

```
    <meta charset="UTF-8">
    <title>Consolas</title>
</head>
<body>
    <table>
        <thead>
            <tr>
                <th>Consola</th>
                <th>Año de lanzamiento</th>
                <th>Año de retirada</th>
            </tr>
        </thead>
        <tbody>
            <tr>
                <td>PlayStation 1</td>
                <td>1994</td>
                <td>2000</td>
            </tr>
            <tr>
                <td>PlayStation 2</td>
                <td>2000</td>
                <td>2004</td>
            </tr>
            <tr>
                <td>PlayStation 3</td>
                <td>2006</td>
                <td>2010</td>
            </tr>
            <tr>
                <td>PS4</td>
                <td>2013</td>
                <td>2017</td>
            </tr>
            <tr>
                <td>PS5</td>
                <td>2020</td>
                <td>En venta</td>
            </tr>
        </tbody>
    </table>
</body>
</html>
```

Actividad propuesta 2.15

Creación de una tabla

Crea un documento HTML que contenga una tabla. Debe estar compuesta por una fila de encabezado y por 3 filas de datos y 4 columnas.

Cada columna se corresponde con un trimestre del año, mostrándose en la fila del encabezado los textos: «1T», «2T», «3T» y «4T».

La primera fila de datos contiene los valores: 1500, 3000, 1800 y 100. La segunda fila contiene los valores 800, 1000, –200 y 80.

Utiliza los elementos <table>, <tr>, <td>, <th> <thead> y <tbody>.

© Ediciones Paraninfo

2.5. Formularios en HTML: estructura y elementos

Un formulario es un bloque de un documento HTML compuesto por un conjunto de elementos de entrada de información. Permite al usuario introducir datos con el objetivo de enviarlos a un servidor para realizar algún tipo de proceso con ellos.

En la web, los formularios son muy habituales; cualquier acceso mediante usuario y contraseña, la búsqueda de artículos en una tienda electrónica, la reserva de la habitación de un hotel o la consulta de la información meteorológica se realizan a través de un formulario. Es uno de los componentes más importantes en la interacción entre los usuarios y los sitios web.

Los formularios se delimitan por la etiqueta <form> y solo tienen utilidad sin contienen elementos de entrada de datos en su interior, ya que estos permitirán a los usuarios introducir la información que se debe enviar al servidor. En la construcción de formularios, por lo tanto, participan un importante número de elementos.

2.5.1. Elementos de los formularios

Un formulario es un contenedor de elementos de entrada de datos. Estos elementos tienen como característica común que están orientados a permitir a los usuarios introducir información para, normalmente, enviarla a un servidor y que sea debidamente tratada. No obstante, estos elementos se pueden utilizar fuera del contexto de los formularios, ya sea para presentar información o para realizar algún tipo de programación mediante el uso de JavaScript.

<form>

Descripción: delimita una sección de una página HTML que contiene los componentes interactivos (formulario) que permiten al usuario introducir y enviar información a un servidor web.

Los formularios en HTML tienen un comportamiento específico y bien definido. Los componentes interactivos posibilitan introducir información y que esta se envíe al servidor, pero se han de cumplir algunas reglas sencillas.

En primer lugar, hay que definir el destino (atributo **action**) y el modo de envío de los datos (atributo **method**). En segundo lugar, hay que indicar de qué componentes queremos enviar los valores introducidos por el usuario (atributo **name**). Por último, hay que especificar cuándo se quiere hacer el envío (elemento **<input>** de tipo **submit**). Opcionalmente, se pueden incluir reglas para validar los datos de entrada y estas se validarán de forma automática antes del envío, cancelándose si dichas reglas no se cumplen.

Por lo tanto, el ciclo de funcionamiento de un formulario es el siguiente:

- El usuario introduce los datos en los campos de entrada correspondientes.
- El usuario pulsa el botón de envío (**submit**).

© Ediciones Paraninfo

- El navegador valida las reglas asignadas a los campos de entrada.
- Si las reglas se cumplen, el navegador prepara los datos y los envía al destino indicado según el método de envío seleccionado.

A continuación se muestra un formulario de ejemplo. En él se indica que el destinatario de los datos introducidos por el usuario en un programa escrito en lenguaje PHP ubicado en un servidor local, que el método de envío es GET y que los datos que se van a enviar son los que se introduzcan en los elementos **<input>** que tienen atributo **name.**

```
<form action="http://localhost/calculadoraimpuestos.php" method="GET">
    <input type="text" placeholder="Concepto" name="concepto">
    <input type="number" placeholder="Importe" name="importe">
    <input type="submit" value="Calcular">
</form>
```

En el navegador, el formulario anterior se muestra tal y cómo se puede observar en la Figura 2.51.

Figura 2.51. *El atributo* placeholder *permite mostrar un texto informativo en el campo* input.

En este mismo formulario se pueden incluir unas reglas que «obliguen» al usuario a introducir valores en los campos de entrada. Para ello se utiliza el atributo **required.**

```
<form action="http://localhost/calculadoraimpuestos.php" method="GET">
    <input type="text" placeholder="Concepto" name="concepto" required>
    <input type="number" placeholder="Importe" name="importe" required>
    <input type="submit" value="Calcular">
</form>
```

Con esta modificación, al pulsar el botón de envío, el navegador verifica que los campos de entrada se han rellenado, cancelando el envío y mostrando un mensaje de error.

Figura 2.52. *Las validaciones de los campos se realizan al enviar los datos del formulario.*

En la declaración de un elemento **<form>** participan una serie de atributos de gran importancia, ya que determinan cómo se va a comportar el formulario. Gracias a estos atributos se puede indicar el destino de los datos, el método de envío o la pestaña del navegador en la que se van a cargar. En la Tabla 2.5, se muestran los atributos más importantes.

© Ediciones Paraninfo

Tabla 2.5. Principales atributos del elemento <form>

Atributo	Valores más importantes	Descripción
action	**URL de destino**	**Determina la URL (normalmente la dirección de un programa en un servidor web) que va a procesar el formulario.**
method	**GET, POST**	**Determina el método de envío. Con valor GET, los datos se envían como un anexo a la URL (separados de esta por el símbolo de cierre de interrogación «?»). Los datos serán visibles en el campo de dirección del navegador. Con POST, los datos se envían en el cuerpo de la petición y, por lo tanto, no son visibles en una navegación convencional.**
target	**_self (por defecto), _blank**	**Indica dónde mostrar el contenido devuelto por el servidor tras el envío del formulario. «_self» lo muestra en el mismo espacio (normalmente en la misma pestaña) en la que se mostró el formulario; «_blank» en un nuevo espacio (normalmente una nueva pestaña).**
autocomplete	**on, off**	**Indica si se deben autocompletar los campos con los valores introducidos en ejecuciones anteriores.**

Nota técnica

Los datos enviados con los métodos POST y GET siempre pueden verse desde el navegador. En el caso del método GET, estos se pueden observar a simple vista, ya que se muestran en la barra de direcciones del navegador. En el envío mediante el método POST hay que acceder a las «Herramientas para desarrolladores», incluidas en todos los navegadores, y en las que se pueden consultar los datos intercambiados entre el navegador y el servidor, entre ellos, los correspondientes a los formularios. Incluso los valores de las contraseñas introducidos en elementos **<input>** de tipo **password** son visibles, ya que este tipo de elemento de entrada de datos solo es una máscara gráfica y no realiza ningún tipo de cifrado ni codificación.

<input>

Descripción: permite proporcionar un componente interactivo para que el usuario introduzca información para su envío a un servidor web. Es una etiqueta que contiene una enorme variedad de tipos de entrada, capaces de permitir introducir desde un texto simple hasta un color, pasando por fechas, contraseñas o valores numéricos dentro de un rango.

Los elementos **<input>** siempre recogen una cadena de caracteres con el objetivo de enviársela al servidor, independientemente del tipo de entrada que representen. El hecho de que haya tantos tipos permite proporcionar al navegador una funcionalidad más avanzada. Por una parte, ayuda en la introducción de datos, ya que los diferentes tipos de **<input>** utilizan máscaras y componentes gráficos de entrada que ayudan al usuario en

© Ediciones Paraninfo

la introducción de los datos. Por ejemplo, el tipo «**color**» proporciona un selector de color visual intuitivo. Sin ninguna duda, esta forma de elegir un color es mucho más sencilla que especificar el código RGB correspondiente de manera manual y sin ayuda.

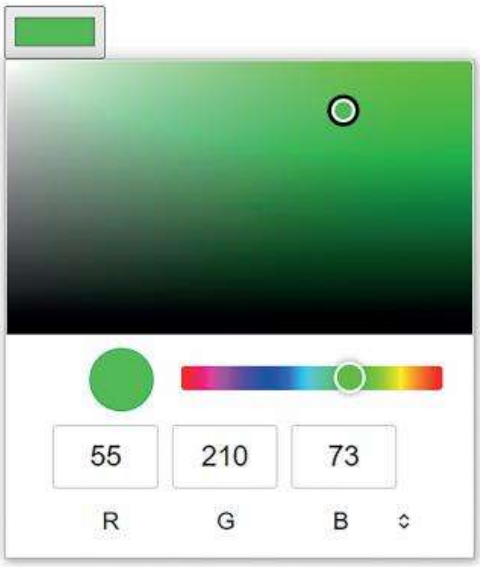


Figura 2.53. *Existen varios sistemas de color. RGB es uno de ellos.*

Nota técnica

RGB son las siglas en inglés de *Red, Green and Blue* (rojo, verde y azul). Es un modelo de color de los denominados **aditivos,** en el que cada color se representa especificando una intensidad para cada uno de los colores primarios.

Cada componente puede tomar un valor entre 0 y 255 (2^8 combinaciones) y se asignan los valores de forma ordenada al rojo (R), verde (G) y azul (B).

Por ejemplo, la combinación de valores 255, 0, 0 representa el color rojo; la combinación 0, 255, 0, el verde; la combinación 0, 0, 255, el color azul; la combinación 0, 0, 0, el negro; la 255, 255, 255, el blanco, y cualquier combinación con los tres valores iguales generará un tono de color gris.

Por otra parte, los componentes aportan seguridad, ya que contienen restricciones y validaciones para certificar que el tipo de dato que se está introduciendo es acorde al tipo de componente diseñado. Por ejemplo, el tipo «**email**» incluye una validación que comprueba, antes de enviar los datos del formulario al servidor, que la dirección de correo electrónico introducida es correcta.

Figura 2.54. *En función del tipo asignado al elemento <input> se realizan unas validaciones concretas.*

© Ediciones Paraninfo

Tabla 2.6. Principales valores del atributo «type» del elemento <input>

Tipo	Tipo de entrada	Representación visual
button	Botón de acción.	Activar
checkbox	Casilla para la selección de un elemento dentro de un conjunto de opciones.	
color	Color.	
date	Fecha.	03/09/2025
datetime-local	Fecha y hora.	24/12/2028 15:12
email	Dirección de correo electrónico.	info@paraninfo.es
file	Selección de un fichero del sistema de archivos.	Seleccionar archivo Lenguaje de marcas.docx
hidden	No es visible pero internamente existe a todos los efectos.	
image	Tiene el mismo comportamiento que el componente **<input>** de tipo **submit,** pero en lugar de texto muestra una imagen.	
month	Mes y año.	septiembre de 1971
number	Valor numérico.	35000
password	Contraseña. Visualmente sustituye los caracteres introducidos por asteriscos o puntos.	••••••
radio	Casilla para la selección de un único elemento dentro de un conjunto de opciones.	
range	Selección visual de un valor numérico dentro de un rango.	
reset	Restaura el estado inicial de los campos del formulario.	Restablecer

© Ediciones Paraninfo

Tipo	Tipo de entrada	Representación visual
search	Texto para realizar una búsqueda. Tiene el mismo comportamiento que el elemento **<input>** de tipo **text,** pero permite establecer una diferenciación para realizar un tratamiento distinto.	
submit	Verifica y envía los datos del formulario al destino indicado en el atributo **action** de este.	Enviar
tel	Número de teléfono.	+34630001100558899
text	Texto libre.	
time	Hora.	18:15
url	URL.	http://www.paraninfo.es
week	Año y ordinal de la semana de dicho año.	Semana 30, 2032

Atributos comunes:

La lista de atributos del componente **<input>** es muy extensa, ya que cada uno de los tipos tiene unos atributos específicos que determinan qué valores se pueden introducir y qué validaciones se deben realizar. Debido a limitación propia de cualquier texto, la lista de atributos significativos de este componente se ha sustituido por la lista de atributos comunes a todos los tipos o a un gran número de ellos. Se recomienda al lector profundizar en los atributos de cada uno de los tipos para poder conocer en detalle el componente **<input>**.

- **autocomplete:** permite activar el autocompletado del componente.
- **autofocus:** el componente que tiene este atributo obtiene el foco cuando se carga la página. Solo hay que utilizarlo, por lo tanto, en un elemento por página.
- **disabled:** desactiva el componente.
- **form:** contiene la referencia al id de un formulario, permitiendo que el componente pertenezca al formulario referenciado, aunque no sea un descendiente de este.
- **list:** permite asociar el componente con una lista de opciones definida en un elemento **<datalist>**. Válido para casi todos los tipos.
- **name:** nombre del elemento de cara a su envío al servidor. Es importante recordar que si el elemento no tiene este atributo, no será enviado, aunque esté contenido por el formulario.

© Ediciones Paraninfo

- **placeholder:** pese a que no es común en todos los tipos de elemento <input>, este atributo está presente en bastantes de ellos. Recibe como valor un texto y este se muestra en el espacio destinado a la entrada de datos. Desaparece cuando el usuario introduce el valor. Permite indicar al usuario qué tipo de dato espera el elemento.
- **readonly:** hace que el valor (atributo **value**) sea visible pero no editable. Válido para casi todos los tipos.
- **required:** convierte el campo en obligatorio. De no rellenarse se mostrará un mensaje de error y la acción de envío de los datos del formulario al servidor no se realizará. Válido para casi todos los tipos.
- **type:** determina la apariencia del componente, el tipo de dato que se va a introducir y la validación. No es obligatorio. Si se omite, el tipo seleccionado por defecto es **text.**
- **value:** valor por defecto del componente.

Actividad propuesta 2.16

Creación de formulario de identificación

Crea un documento HTML que contenga un formulario de identificación en un portal web.

Los datos de entrada son: identificador (un campo de texto), una contraseña (un campo de tipo password) y un botón de verificación y envío.

<button>

Descripción: permite crear un botón. Este botón no tiene una utilidad predefinida como, por ejemplo, los elementos **<input>** de tipo **submit** o **reset**, sino que deben programarse. No necesariamente han de ser utilizados dentro de un formulario.

Ejemplo:

```
<button>Enviar</button>
```

Calcular

Figura 2.55. *Se puede modificar el comportamiento de un botón utilizando JavaScript.*

<select>

Descripción: permite al usuario elegir una opción de un conjunto de ítems mostrado como una lista de opciones (puede mostrarse como lista desplegable). Las opciones se establecen con el componente **<option>** y se pueden agrupar con el componente **<optgroup>**.

© Ediciones Paraninfo

Atributos significativos:

- **multiple:** indica si admite selección múltiple.
- **size:** indica el número de líneas que se mostrarán simultáneamente.

Ejemplo 1:

En este primer ejemplo, al no modificarse el comportamiento por defecto del elemento **<select>**, se muestra una lista plegada con la primera de las opciones disponibles. Al pulsar sobre el componente se despliega la lista permitiendo seleccionar la opción deseada.

```
<select>
    <option>Baloncesto</option>
    <option>Fútbol</option>
    <option>Tenis</option>
    <option>Golf</option>
    <option>Esquí</option>
    <option>Remo</option>
</select>
```

Figura 2.56. *Las listas desplegables ocupan poco espacio.*

Figura 2.57. *Las listas desplegables son más útiles con pocos elementos.*

Ejemplo 2:

En este segundo ejemplo el atributo **size** tiene como valor «4», lo que provoca que aparezca una lista con cuatro elementos visibles. Por otra parte, el atributo **multiple** permite seleccionar múltiples opciones pulsando la tecla **Ctrl** y el botón izquierdo del ratón para hacer selecciones y deselecciones alternas, o pulsando la tecla **Shift** y el botón izquierdo del ratón para hacer selecciones y deselecciones por bloques.

```
<select multiple size="4">
    <option>Baloncesto</option>
    <option>Fútbol</option>
    <option>Tenis</option>
    <option>Golf</option>
    <option>Esquí</option>
    <option>Remo</option>
</select>
```


© Ediciones Paraninfo

Figura 2.58. *Las listas desplegadas ocupan más espacio, pero muestran más información.*

<option>

Descripción: este elemento define una opción seleccionable dentro de un elemento **<select>** o una sugerencia de un elemento **<datalist>.**

Atributos significativos:

- **disabled:** indica si la opción se encuentra deshabilitada.
- **label:** etiqueta la opción y se muestra solo en el caso de que no haya contenido en el elemento.
- **selected:** indica que la opción debe estar seleccionada al cargar el documento.
- **value:** es el dato que se envía al servidor cuando se envía el formulario. No hay que olvidar que el contenido del elemento **<option>** tiene como único objetivo ser presentado en el navegador.

Ejemplo:

```
<select>
    <option value="opcion1">Seat</option>
    <option value="opcion2">Kia</option>
    <option value="opcion3" selected>Renault</option>
    <option value="opcion4" disabled>Ferrari</option>
    <option value="opcion5">Volvo</option>
</select>
```

Figura 2.59. *Los elementos <option> se pueden seleccionar por defecto y desactivar.*

<optgroup>

Descripción: permite agrupar un conjunto de elementos **<option>** de un elemento **<select>.**

© Ediciones Paraninfo

Atributos significativos:

- **disabled:** impide seleccionar la opción.
- **label:** determina el nombre del grupo de opciones.

Ejemplo:

```
<select>
    <optgroup label="Disponibles">
        <option value="opcion1">Seat</option>
        <option value="opcion2">Kia</option>
    </optgroup>
    <optgroup label="Sin stock">
        <option value="opcion4">Ferrari</option>
        <option value="opcion5">Volvo</option>
    </optgroup>
</select>
```

Figura 2.60. *Si el número de elementos es grande, resulta conveniente agruparlos.*

Actividad propuesta 2.17

Creación de una lista de selección

Crea una lista de selección de destinos turísticos agrupados por continentes. Las opciones de la lista de destinos son:

- En Europa: París, Berlín, Roma y Ámsterdam.
- En América: Río de Janeiro, La Habana, Punta Cana y Nueva York.
- En África: Sudáfrica, Kenia, Tanzania y Egipto.
- En Asia: Vietnam, Japón, China y Tailandia.
- En Oceanía: Islas Fiyi, Australia y Nueva Zelanda.

<datalist>

Descripción: define un conjunto de elementos **<option>** como una lista para su uso en otros elementos del formulario. Habitualmente en elementos **<input>** de tipo **text**, aparece como una lista de opciones al pulsar sobre el componente o genera una ayuda contextual filtrando las opciones según se escriben los caracteres en la caja de texto.

© Ediciones Paraninfo

Ejemplo:

Es importante observar que el elemento **<datalist>** contiene un atributo «**id**» y que es referenciado por parte del atributo «**list**» del componente **<input>**. Este es el mecanismo para relacionar un componente de entrada con la lista de datos.

```
<input type="text" list="lenguajes" placeholder="Selecciona un lenguaje">
<datalist id="lenguajes">
    <option>Java</option>
    <option>C#</option>
    <option>Cobol</option>
    <option>Pascal</option>
    <option>Python</option>
</datalist>
```

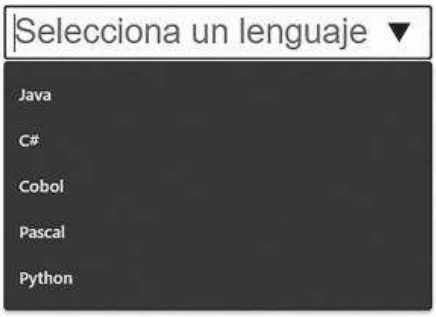


Figura 2.61. *Utilizando* datalist *se puede reutilizar parte del código HTML.*

<fieldset>

Descripción: permite agrupar un conjunto de campos de un formulario. La agrupación se realiza de forma visual, mediante un marco, y de forma funcional, permitiendo realizar acciones sobre todos los elementos agrupados. Mediante el elemento **<legend>** se puede asignar un texto que haga de título del marco.

Atributos significativos:

- **disabled:** permite deshabilitar todos los elementos agrupados.
- **form:** el valor del atributo **id** del formulario al que está asociado.
- **name:** asigna un nombre al conjunto de campos.

Ejemplo:

```
<fieldset>
    <legend>Cliente particular:</legend>
    <p>Nombre: <input type="text" placeholder="Nombre"></p>
    <p>Dirección: <input type="text" placeholder="Dirección de envío"></p>
</fieldset>
```

© Ediciones Paraninfo

```
<fieldset disabled>
<legend>Cliente empresa:</legend>
    <p>Nombre: <input type="text" placeholder="Nombre de la empresa"></p>
    <p>Dirección: <input type="text" placeholder="Dirección de envío"></p>
    <p>CIF: <input type="text" placeholder="Código de Identificación
Fiscal"></p>
</fieldset>
```

Figura 2.62. *Los elementos* fieldset *mejoran la legibilidad de los formularios.*

<legend>

Descripción: permite asignar un título o leyenda a un conjunto de campos agrupados mediante un elemento **<fieldset>**.

<label>

Descripción: identifica una etiqueta de un campo de entrada de datos del formulario. También permite poner el foco en el campo de entrada pulsando sobre el texto de la etiqueta. La asociación se puede realizar de dos maneras: conteniendo el campo de entrada entre las etiquetas de apertura y cierre del elemento **<label>** o utilizando el atributo **for** cuyo valor deberá ser el del atributo **id** del componente del formulario relacionado.

Atributos significativos:

- **for:** contiene la referencia al componente con el que está asociado el elemento **<label>**.

Ejemplo:

```
<p>
<label for="apodo">Apodo:</label>
<input type="text" id="apodo">
```


© Ediciones Paraninfo

```
</p>
<p>
    <label>
        Dirección de correo:
        <input type="email">
    </label>
</p>
```

Apodo:

Dirección de correo:

Figura 2.63. *El elemento* label *aporta mejoras en la usabilidad.*

<meter>

Descripción: representa un valor dentro de un rango. Resulta útil para representar calificaciones, resultados en encuestas, grado de ocupación de un recurso, etc. Para medir grados de avance es más apropiado utilizar el componente **<progress>.**

En ausencia de los atributos **low, high** y **optimum,** todas las barras se representarán en color verde. Utilizando los atributos **low, high** y **optimum,** se establecerán tres segmentos: **min-low, low-high** y **high-max.** Si el valor de **optimum** está en el **último** segmento, el color de la barra para un **value** que esté dentro de este se coloreará en verde. Si el **value** está en el segmento anterior, se coloreará en naranja. Por **último,** si el **value** está en el primer segmento, se coloreará en rojo.

Atributos significativos:

- **value:** indica el valor actual.
- **min:** indica el valor mínimo.
- **max:** indica el valor máximo.
- **low:** indica el umbral por debajo del cual se considera que el valor es bajo.
- **high:** indica el umbral por encima del cual se considera que el valor es alto.
- **optimum:** indica el valor considerado óptimo.

Ejemplo 1:

```
<p>Avance del proceso: <meter min="0" max="100" value="20"></meter></p>
<p>Avance del proceso: <meter min="0" max="100" value="90"></meter></p>
```

Avance del proceso:

Avance del proceso:

Figura 2.64. *Es conveniente mantener informado a los usuarios del avance de los procesos.*

© Ediciones Paraninfo

Ejemplo 2:

```
<p>Calificación: <meter min="0" max="10" low="5" high="7" optimum="8"
value="4.9"></meter></p>
<p>Calificación: <meter min="0" max="10" low="5" high="7" optimum="8"
value="5"></meter></p>
<p>Calificación: <meter min="0" max="10" low="5" high="7" optimum="8"
value="6.9"></meter></p>
<p>Calificación: <meter min="0" max="10" low="5" high="7" optimum="8"
value="7"></meter></p>
<p>Calificación: <meter min="0" max="10" low="5" high="7" optimum="8"
value="10"></meter></p>
```

Figura 2.65. *Los códigos de colores facilitan la presentación de determinados datos.*

<output>

Descripción: delimita el resultado de una operación realizada, normalmente, por un programa.

Atributos significativos:

- **for:** contiene la relación separada por espacios de los identificadores de los elementos que participan en el cálculo. Es información semántica.
- **form:** incluye el identificador del formulario con el que está asociado.
- **name:** nombre con el que el valor del campo se envía al servidor.

Ejemplo:

```
<input type="number" id="operador1" value="8">
+
<input type="number" id="operador2" value="2">
=
<output for="operador1 operador2" name="resultado">10</output>
```


© Ediciones Paraninfo

8 + 2 = 10

Figura 2.66. *No se deben utilizar elementos* input *para mostrar datos de salida.*

<progress>

Descripción: muestra un indicador visual del grado de avance de un proceso. Si no se le asigna el atributo **value**, el indicador de avance aparece en movimiento de izquierda a derecha y de derecha a izquierda, informando de que el grado de avance del proceso es desconocido.

No se debe confundir con el componente **<meter>**.

Atributos significativos:

- **value:** valor actual.
- **max:** valor máximo.

Ejemplo:

```
<progress></progress>
<progress max="10" value="3"></progress>
```

Figura 2.67. *Si es posible, es conveniente mostrar el grado de avance, ya que aporta más información.*

<textarea>

Descripción: proporciona un área para la introducción de texto. La principal diferencia con el elemento **<input>** de tipo **text** es que **<textarea>** permite introducir múltiples líneas. En la esquina inferior derecha dispone de un selector para cambiar el tamaño del componente con el ratón. Esta funcionalidad se puede desactivar mediante estilos CSS.

Atributos significativos:

- **cols:** determina el ancho del componente.
- **maxlength:** indica el número máximo de caracteres que admite.
- **minlength:** señala el número mínimo de caracteres que admite.
- **placeholder:** texto mostrado en el componente como ayuda para el usuario. Desaparece cuando se le asigna un valor al elemento.
- **readonly:** indica que el componente solo es de lectura.
- **required:** hace obligatorio introducir valor en el componente para cumplir con las reglas de validación.

© Ediciones Paraninfo

- **rows:** número de líneas visibles. Si el texto excede el espacio disponible, se muestra una barra de desplazamiento vertical.
- **spellcheck:** determina si el texto debe ser revisado ortográfica y gramaticalmente. Admite los valores **true** y **false.**
- **wrap:** indica el método de ajuste del texto para el envío al servidor. Admite los valores **soft** y **hard.** Visualmente ambos métodos se presentan igual, rompiendo las líneas para que se vean por completo. Internamente, el método **hard** envía dichos saltos de línea como parte del texto, mientas que el método **soft** envía el texto tal cual ha sido introducido.

Ejemplo:

```
<textarea name="hard" cols="40" rows="8" wrap="hard">

</textarea>
```

```
    Lorem ipsum dolor, sit amet
consectetur adipisicing elit. Aliquam
accusamus reiciendis modi aliquid soluta
cumque impedit officia omnis labore? Qui
id debitis minus atque necessitatibus
quos? Eligendi sapiente tempora
distinctio!
    Corporis veritatis placeat nihil
```

Figura 2.68. *El elemento* textarea *se debe utilizar para introducir bloques de texto.*

Actividad propuesta 2.18

Creación de un formulario de contratación de servicios

Crea un documento HTML que incluya un formulario para la contratación de una línea de telefonía móvil. Los datos que se van a solicitar son:

- Nombre del usuario, dirección y número de cuenta bancaria.
- Tarifa. Se debe mostrar una lista desplegable con las opciones «Tarifa Joven», «Tarifa Sénior» y «Tarifa Plana Superior». Solo se podrá seleccionar una de las opciones.
- Fecha de comienzo del servicio, utilizando un calendario.
- Notificaciones por SMS. Se ofrecerán dos campos **<input>** de tipo **radio** con las opciones «Sí» y «No», pudiéndose elegir solo una de ellas.
- Un botón de envío del formulario y un botón de reinicio del formulario.

2.5.2. Validaciones

Antes de enviar un formulario al servidor es conveniente validar que los datos introducidos por el usuario cumplen con las restricciones que se hayan establecido. Por ejemplo, algu-

© Ediciones Paraninfo

nos campos de entrada serán obligatorios, los valores de determinados campos tendrán que estar dentro de los rangos definidos y la secuencia de caracteres introducida deberá seguir un patrón concreto.

Durante la definición de un formulario de entrada de datos, se especifican las condiciones que deben cumplir dichos datos, siendo esta una tarea ligada íntimamente a la fase de diseño.

La validación de las reglas se realiza en una fase previa al envío del formulario mediante la pulsación de un elemento **<input>** de tipo **submit.** Alternativamente se pueden programar las reglas de validación en JavaScript, pero esta opción siempre requiere más esfuerzo y deberá elegirse solo en caso de necesidad.

Todos los componentes de entrada de datos tienen atributos para definir reglas de validación. Algunos de los atributos referentes a la validación de formularios son los siguientes:

- **required:** indica que el campo de entrada es obligatorio. Hay que tener presente que, en los campos de entrada de tipo texto, los espacios en blanco ya se consideran como valores válidos.
- **minlength y maxlength:** determinan las longitudes mínimas y máximas en campos de entrada de tipo texto.
- **min y max:** establecen los valores mínimos y máximos en campos de entrada numéricos.
- **type:** en función del valor asignado a este parámetro, se fijan los valores correctos de entrada. Por ejemplo, si el valor es **email,** se validará que el valor introducido es una dirección de correo electrónico.
- **pattern:** comprueba que el valor introducido es acorde a una **expresión regular.**

Una expresión regular es un patrón formado por una cadena de caracteres que define una regla. Es un mecanismo extremadamente flexible, pero que tiene un grado de complejidad relativamente alto. No está dentro del objetivo de este libro profundizar en las expresiones regulares, aunque merece la pena ilustrar con un ejemplo en qué consisten.

Ejemplo:

Dado un campo de entrada **<input>** de tipo **text,** se desea que el valor introducido esté formado por una secuencia de caracteres formado por **un dígito numérico** seguido de **una letra minúscula** y **finalizado por el carácter #.** La expresión regular capaz de validar esta regla será la siguiente:

\d[a-z]#

Donde:

- \d indica que se debe introducir un dígito numérico.
- [a-z] señala que se debe introducir una letra minúscula entre la letra a y la letra z.
- # informa de que hay que introducir el símbolo de la almohadilla (#).

Valores válidos para esta expresión serán, por ejemplo: **1f#**, **8i#** y **0w#.**

© Ediciones Paraninfo

En un formulario HTML, se puede incorporar esta validación en un campo **<input>** de tipo **text** de la siguiente manera:

```
<form action="">
    <input type="text" name="comando" pattern="\d[a-z]#">
    <input type="submit" value="Enviar">
</form>
```

Nota técnica

Una aplicación web consta de una parte denominada **cliente** (se ejecuta en el navegador) y otra parte denominada **servidor** (se ejecuta, como indica su nombre, en el ordenador servidor). Las validaciones se pueden realizar en ambas partes de la aplicación. En este libro las validaciones que se presentan son las correspondientes a la parte cliente.

Es muy importante saber que las validaciones realizadas en el lado cliente son relativamente sencillas de «puentear» si se dispone de unos conocimientos técnicos intermedios sobre el desarrollo de aplicaciones web.

En caso de desarrollar una aplicación web con un nivel de fiabilidad medio o medio-alto, es conveniente disponer de validación de datos en la parte ubicada en el servidor, ya que esta es mucho más difícil de evitar.

Actividad resuelta 2.4

Formulario de registro

Crear un documento HTML que contenga un formulario de registro a una tienda online. Deberán introducirse obligatoriamente los campos **nombre** y **dirección de correo electrónico.** El usuario podrá elegir si desea suscribirse a las notificaciones por correo electrónico o no. Los datos se deberán enviar usando el método POST a **www.servidorlocal.edu/programa.php**.

Solución

```
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <title>Actividad Resuelta</title>
</head>
<body>
    <form method="POST" action="http://www.servidorlocal.edu/programa.php">
        <label for="nombre">Nombre:</label>
        <input type="text" name="nombre" id="nombre" required>
        <br><br>
        <label for="correo">Dirección de correo electrónico:</label>
        <input type="email" name="correo" id="correo" required>
        <br><br>
        <label for="recibir">Desea recibir notificaciones:</label>
        <input type="checkbox" name="recibir" id="recibir">
```


© Ediciones Paraninfo

```
        <br><br>
        <input type="submit" value="Enviar">
    </form>
</body>
</html>
```

Nota sobre la solución

La separación entre las distintas partes del formulario se ha realizado utilizando elementos <br>. Existen mejores soluciones, pero estas pasan por la utilización de estilos CSS. Debido a que los estilos CSS se tratan en la Unidad 3, se ha preferido elegir una solución para el espaciado entre las partes del formulario menos correcta pero más comprensible en este momento del aprendizaje.

Actividad propuesta 2.19

Creación de un formulario de registro

Crea un documento HTML que contenga un formulario de registro en un portal web.

Los datos de entrada son:

- Identificador (una dirección de correo electrónica válida).
- Alias del usuario (un campo de texto de una longitud máxima de 10 caracteres).
- Contraseña (un campo de tipo password de una longitud máxima de 8 caracteres).
- Botón de envío.

Todos estos campos son obligatorios.

2.6. Multimedia en HTML

En HTML existe un breve pero potente conjunto de elementos multimedia. Gracias a ellos se puede componer una página que incluya imágenes estáticas, vídeos, mapas interactivos, música, sonidos, animaciones o videojuegos.

Los elementos multimedia han estado presentes en la web desde el principio mediante los elementos <img>. Durante esos primeros años las redes de comunicaciones habituales eran muy lentas y se debía optimizar su uso: se limitaba el tamaño de las imágenes al mínimo de resolución y tamaño necesario para su correcta visualización. Por suerte, las redes de comunicaciones han mejorado de forma drástica y todas esas restricciones prácticamente han desaparecido o se han relajado. En la actualidad es habitual ver contenido multimedia en alta resolución y emitido en directo *(streaming)* en casi cualquier hogar.

HTML5 incrementó la capacidad de incluir multimedia en las páginas web mediante los componentes <audio> y <video>, proporcionando nuevas oportunidades para los diseñadores y programadores de páginas.

© Ediciones Paraninfo

2.6.1. Elementos multimedia

Con HTML5 se introdujo un interesante conjunto de elementos para incluir contenidos multimedia a las páginas web, evitando utilizar soluciones no estandarizadas que pueden provocar problemas de compatibilidad. Estos elementos son muy completos y, a la vez, sencillos de utilizar. También disponen de propiedades para adaptar los contenidos a los diferentes dispositivos, así como dotarlos de la capacidad de ser accesibles para personas con discapacidad.

<audio>

Descripción: permite insertar recursos de sonido en la página. Es un componente muy sencillo de utilizar y está soportado por la gran mayoría de navegadores. Para establecer el recurso que reproducir se puede asignar mediante el atributo **src** o con el elemento **<source>**.

Admite proporcionar un contenido alternativo por si el recurso no es compatible con el navegador.

Atributos significativos:

- **src:** determina la URI del recurso.
- **preload:** indica al navegador cómo se desea que se haga la precarga del recurso. Los valores posibles son **none** (no se precarga el recurso hasta que se vaya a utilizar minimizando el tráfico de datos), **metadata** (se precargan los metadatos del recurso) o **auto** (se deja a criterio del navegador).
- **autoplay:** indica al navegador que comience la reproducción automática del recurso tan pronto le sea posible. Debido a que la reproducción automática de sonido puede resultar una experiencia negativa para el usuario, la funcionalidad de reproducción automática está sujeta a una serie de condiciones adicionales.
- **loop:** indica que se desea realizar la reproducción en bucle.
- **muted:** si existe, silencia el componente.
- **controls:** si existe, se proporciona una interfaz para que el usuario pueda controlar la reproducción.

Ejemplo:

```
<audio src="rugido.ogg" controls loop>
Este contenido no es compatible con el navegador.
</audio>
```

Figura 2.69. *El reproductor de audio puede tener distinto aspecto en diferentes navegadores.*

© Ediciones Paraninfo

Actividad propuesta 2.20

Reproductor de audio

Crea un documento HTML que contenga un reproductor de audio que haga referencia a un fichero de sonido alojado en un servidor externo. El reproductor debe mostrar los controles de manejo del fichero.

En el portal Wikipedia Commons existen múltiples recursos multimedia que es posible referenciar en proyectos personales. Un ejemplo que se puede utilizar para realizar esta actividad es la grabación de la conversación que se mantuvo entre los astronautas de la NASA y el centro de control durante el primer alunizaje. El fichero se encuentra alojado en el siguiente enlace:

https://upload.wikimedia.org/wikipedia/commons/e/e5/Pouso_da_Apollo_11_na_Lua.ogg

<video>

Descripción: permite insertar recursos de vídeo en la página. Es un componente muy sencillo de utilizar y está soportado por la gran mayoría de navegadores. Para establecer el recurso que reproducir se puede asignar mediante el atributo **src** o con el elemento **<source>.** Permite proporcionar un contenido alternativo por si el recurso no es compatible con el navegador.

Atributos significativos:

- **src:** determina la URI del recurso.
- **poster:** establece la URI de una imagen para mostrar mientras se carga o inicia la reproducción del vídeo.
- **playsinline:** muestra el vídeo dentro de su área de reproducción en lugar de hacerlo a pantalla completa. Normalmente los navegadores tienen este comportamiento como predeterminado, por lo que, en esos casos, el atributo no tendrá efecto.
- **preload:** indica al navegador cómo se desea que se haga la precarga del recurso. Los valores posibles son **none** (no se precarga el recurso hasta que se vaya a utilizar minimizando el tráfico de datos), **metadata** (se precargan los metadatos del recurso) o **auto** (se deja a criterio del navegador).
- **autoplay:** indica al navegador que comience la reproducción automática del recurso tan pronto le sea posible. Debido a que la reproducción automática de vídeos puede resultar una experiencia negativa para el usuario, la funcionalidad de reproducción automática está sujeta a una serie de condiciones adicionales.
- **loop:** señala que se desea realizar la reproducción en bucle.
- **muted:** si existe, silencia el componente.
- **controls:** si existe, se proporciona una interfaz para que el usuario pueda controlar la reproducción.
- **width y height:** determinan el ancho y el alto del reproductor de vídeo.

© Ediciones Paraninfo

Ejemplo:

```
<video src="trailerpelicula.avi" poster="posterpelicula.jpg" width="300"
height="300" controls>
Contenido no soportado
</video>
```

Figura 2.70. *El navegador y sus componentes determinan qué formatos de vídeo se pueden reproducir.*

<track>

Descripción: permite incluir una pista con información sobre los elementos multimedia **<audio>** o **<video>** en formato WebVTT (extensión .vtt). Con este tipo de ficheros se puede almacenar información temporizada como, por ejemplo, los subtítulos.

Atributos significativos:

- **default:** determina el elemento **<track>** que utilizar por defecto.
- **kind:** indica el tipo de pista. Admite las opciones **subtitles** (subtitulado), **captions** (transcripciones), **descriptions** (descripciones), **chapters** (capítulos) y **metadata** (información no visible para el usuario y utilizada por programas).
- **label:** un título que describe la naturaleza de la pista.
- **src:** determina la URI del recurso.
- **srclang:** identifica el lenguaje del recurso.

Ejemplo:

```
<video width="800" controls>
    <source src="fuego.mp4" type="video/mp4">
```


© Ediciones Paraninfo

```
    <source src="fuego.webm" type="video/webm">
    <track label="English" kind="subtitles" srclang="en" src="subtitle.vtt">
    <track label="Español" kind="subtitles" srclang="es" src="subtitulos.vtt"
default>
</video>
```

Figura 2.71. *Los subtítulos posibilitan que personas con discapacidad auditiva accedan a la información.*

<img>

Descripción: permite insertar una imagen en la página HTML a partir de un recurso externo referenciado mediante el atributo **src.**

El estándar no indica cuáles son los formatos que se deben aceptar, por lo que la compatibilidad dependerá del navegador. Los formatos más habituales utilizados en web y soportados por los navegadores más populares son los siguientes:

Tabla 2.7. Formatos gráficos soportados por los navegadores web

Formato	Extensión	Tipo MIME
Animated Portable Network Graphics (APNG)	.apng	image/apng
Graphics Interchange Format (GIF)	.gif	image/gif
Joint Photographic Expert Group image (JPEG)	.jpg, .jpeg, .jpe, .jif, .jfif, .jfi	image/jpeg
Portable Network Graphics (PNG)	.png	image/png
Scalable Vector Graphics	.svg	image/svg+xml
Web Picture format (WebP)	.webp	Image/webp

© Ediciones Paraninfo

Atributos significativos:

- **alt:** contiene un texto con información equivalente a la proporcionada por la imagen. Se utiliza cuando la imagen no puede verse (usuario con discapacidad, formato no soportado por el navegador…).
- **src:** contiene la URI del recurso de imagen que se va a mostrar. Es un atributo obligatorio.
- **srcset y sizes:** permiten asignar varias imágenes junto con indicaciones para ayudar al navegador a elegir la versión más adecuada.
- **usemap:** contiene la referencia a un mapa definido en un elemento <map>.
- **ismap:** si existe, indica que la imagen es parte de un mapa del lado servidor.
- **width y height:** determinan el número de píxeles de ancho y alto, respectivamente, que ocupará la imagen. Si se indica uno de los dos atributos, el otro se calculará de forma automática para mantener la proporción original de la imagen. Si se indican ambos, la imagen se mostrará con el tamaño exacto indicado, pero hay que tener cuidado para no deformar la imagen.
- **decoding:** indica al navegador si la imagen debe ser decodificada de manera síncrona (valor **sync**) o asíncrona (valor **async**). Con el valor **auto** (por defecto), el navegador decide qué método utilizar.
- **loading:** permite indicar al navegador qué estrategia de carga de las imágenes utilizar. Con el valor **eager**, la carga se realiza inmediatamente en todos los casos; con valor **lazy**, la carga se difiere hasta que la sección de página visible en el navegador está próxima a la posición de la imagen.

<map>

Descripción: dado un elemento **<img>**, permite definir áreas de navegación utilizando el elemento **<area>**. Estas áreas contendrán enlaces a otros recursos. Es decir, posibilita incrustar enlaces en regiones de una imagen para construir imágenes interactivas con mapas de navegación.

Atributos significativos:

- **name:** identifica al mapa para poder ser referenciado por el atributo **usemap** del elemento **<img>** que lo utilice.

<area>

Descripción: define una región dentro de un elemento **<map>** permitiendo convertirla en un enlace a otro recurso. Admite varios tipos de figuras (rectángulos, círculos y polígonos) para delimitar el área. Contienen enlaces a otros recursos.

Atributos significativos:

- **alt:** versión textual del área.

© Ediciones Paraninfo

- **shape:** tipo de figura que determina el área. Puede tener los valores **default** (toda el área del mapa), **rect** (una región rectangular), **circle** (una región circular) o **poly** (un polígono).
- **coords:** determina las coordenadas que forman el área. El formato depende del valor de **shape.**
- **href:** la URI al recurso enlazado por el área.
- **target:** ubicación donde ha de abrirse el recurso enlazado. Al igual que el elemento **<a>**, admite los valores **_blank**, **_parent**, **_self** y **_top.**
- **download:** indica si el recurso debe abrirse o descargarse.
- **ping:** listado de URL a las que ha de conectarse el navegador cuando se selecciona el área.
- **rel:** información sobre el recurso enlazado.

Ejemplo:

En este ejemplo se crea un mapa en el que se delimitan dos áreas de navegación: una correspondiente al círculo amarillo y otra al cuadrado rojo. Al pasar el ratón por encima de cualquiera de las dos áreas, cambia el cursor del ratón, indicando que se trata de un enlace. En el elemento **<img>** se establece una relación de uso del mapa mediante el atributo **usemap.**

```
<map name="mapa_seleccion">
    <area href="resultado.html" alt="Activar" shape="circle"
coords="205,187,170">
    <area href="cubo1.html" alt="Cubo superior" shape="rect"
coords="420,30,515,130">
</map>
<img src="mapimage.png" usemap="#mapa_seleccion" alt="Mapa de ejemplo">
```

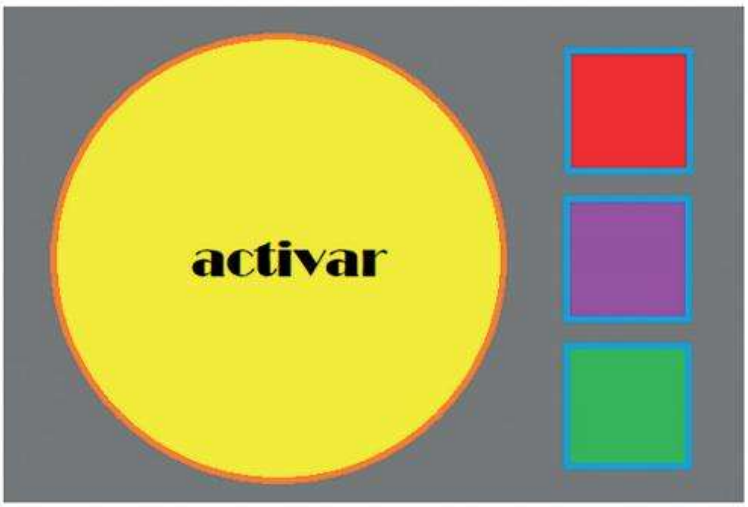


Figura 2.72. *Las imágenes interactivas pueden ser un recurso de navegación.*

© Ediciones Paraninfo

<svg>

Descripción: este elemento permite incrustar un gráfico vectorial en el documento HTML, definiendo su aspecto mediante primitivas expresadas en XML. Su especificación completa está fuera del alcance de este libro, por lo que no se detalla su sintaxis ni estructura.

Ejemplo:

```
<svg xmlns="http://www.w3.org/2000/svg" width="150" height="100" viewBox="0 0
4 3">
  <rect width="4" height="1" y="0" fill="#00ab39" />
  <rect width="4" height="1" y="1" fill="#ffffff" />
  <rect width="4" height="1" y="2" fill="#000000" />
</svg>
```

Figura 2.73. *Existen programas de diseño gráfico que generan código SVG.*

<canvas>

Descripción: proporciona un lienzo sobre el que mostrar gráficos y animaciones mediante programas escritos en JavaScript. Permite crear contenidos interactivos, anuncios animados o videojuegos. Este componente, incorporado en HTML5, se gestiona de manera programática, por lo que la explicación detallada de su uso excede del objetivo de este libro.

Atributos significativos:

- **width:** determina el ancho del componente.
- **height:** determina el alto del componente.

Ejemplo:

```
<canvas id="lienzo" width="500" height="500"></canvas>
```


© Ediciones Paraninfo

2.7. XHTML: la versión XML de HTML

XHTML son las siglas de *eXtensible HyperText Markup Language* («lenguaje de marcado de hipertexto extensible»). XHTML es, esencialmente, la versión XML de HTML, e incluye las mismas funcionalidades. Es más estricto, robusto y está mejor estructurado. Facilita el procesamiento automático y permite realizar validaciones de manera más sencilla. Se definió como una evolución de HTML para avanzar hacia la denominada **web semántica.**

Nota técnica

La web semántica es una iniciativa del W3C para conseguir que los contenidos de la web sean procesables de manera sencilla y fiable por aplicaciones informáticas.

Para conseguirlo, una de las tecnologías desarrolladas fue XHTML, una modificación de HTML, que aporta estructura y rigor, además de permitir incluir información semántica en las páginas web.

Uno de los factores de éxito de HTML y de la web es la flexibilidad con la que los navegadores tratan los errores existentes en los documentos. Por ejemplo, dado el siguiente código:

```
<body>
    <table>
        <tr>
            <td>Uno
            <td>Dos
        <tr>
            <td> Tres
            <td>Cuatro
</body>
```

Pese a que no se han cerrado los elementos **<table>**, **<tr>** ni **<td>**, prácticamente todos los navegadores más utilizados mostrarán el resultado recogido en la Figura 2.74 (la línea de puntos se le ha asignado con un estilo CSS para ilustrar mejor el ejemplo).

Figura 2.74. *XHTML es diferente a HTML en estructura, pero no en capacidad.*

Como se puede observar, los navegadores son muy laxos a la hora de interpretar el código HTML de la página y suelen acertar con la «solución» propuesta al descuido del autor de dicha página.

© Ediciones Paraninfo

Esta aparente ventaja se convierte en inconveniente cuando se desea realizar un procesado automático de la página por parte de un programa informático. No es lo mismo interpretar un error para mostrar una representación visual que va a ser vista e interpretada por una persona que hacerlo para que lo procese un ordenador, ya que este último no tiene un criterio abstracto suficiente para determinar que lo que está analizando no tiene sentido.

XHTML resuelve ese problema. Con la misma capacidad de representación que HTML, incorpora las reglas de XML convirtiendo el lenguaje en una herramienta mucho más robusta.

Pese a sus ventajas y a su robustez, el objetivo inicial de convertir XHTML en la evolución de HTML (en su versión 4) no se ha consolidado, y en la actualidad la combinación de HTML5 y CSS3 se ha convertido en el estándar de tecnologías para el diseño de páginas web.

Los documentos XHTML se almacenan en texto plano en ficheros con la extensión **.xhtml.**

2.7.1. Fundamentos

XHTML está basado en XML, un metalenguaje de marcado ampliamente utilizado en un gran número de contextos y constituido por un breve conjunto de sencillas reglas que lo dotan de una gran **robustez.**

Además de la robustez, tomar el metalenguaje XML como base proporciona a XHTML un buen conjunto de tecnologías periféricas que permiten realizar validaciones o transformaciones útiles para el **procesamiento automático** de los documentos.

En sus versiones iniciales, HTML incluía atributos para modificar la apariencia de los elementos y dotar de mayor vistosidad a las páginas web. Con XHTML se da el primer paso para separar los datos y su estructura (lo que se denomina *modelo de datos*) de la presentación de estos (lo que se denomina *vista*), realizada mediante estilos CSS. XHTML mejora la **estructura** de las páginas, aunque esto también se ha incluido en HTML5.

XHTML es **extensible.** Permite incluir espacios de nombres y ampliar el conjunto de etiquetas que se pueden utilizar más allá de las heredadas de HTML.

2.7.2. Diferencias entre XHTML y HTML

Existen varias diferencias entre XHTML y HTML. Casi todas tienen que ver con el hecho de que XHTML es un documento XML y, por lo tanto, tiene que estar bien formado y ser gramaticalmente correcto. Algunas de las diferencias son las siguientes:

- Todos los elementos deben cerrarse, incluso los vacíos. Los elementos vacíos en HTML no se cerraban. El ejemplo más característico es el elemento **
**, que al tratarse de un elemento vacío (no contiene a otros elementos) no tiene etiqueta de cierre. En XHTML este elemento se debe escribir como **
.**

© Ediciones Paraninfo

- Los elementos no vacíos que no tengan contenido no pueden cerrarse de forma abreviada. Por ejemplo, un elemento ** ** sin contenido no se puede escribir en su forma abreviada (** **), sino con apertura y cierre (** **).
- Los elementos deben estar correctamente anidados.
- XHTML distingue entre mayúsculas y minúsculas. Todas las etiquetas deben estar escritas en minúsculas.
- En XHTML los valores de los atributos deben ir obligatoriamente entre comillas.
- Todos los atributos deben tener asignado un valor. Por ejemplo el atributo **required** de algunos tipos de elemento **<input>** debe expresarse como **required="true"** y no simplemente como **required**, tal y como se hace en HTML5.

2.7.3. Representación de XHTML

Los navegadores interpretan XHTML aportando resultados similares a los proporcionados en las páginas equivalentes en HTML.

El siguiente código es un ejemplo básico de un documento XHTML:

```
<?xml version="1.0" encoding="UTF-8"?>
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="es">
  <head>
    <title>Título de la página</title>
  </head>
  <body>
    <h3>¡HOLA!</h3>
  </body>
</html>
```

El siguiente código es un ejemplo básico equivalente en formato HTML:

```
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <title>Título de la página</title>
</head>
<body>
    <h3>¡HOLA!</h3>
</body>
</html>
```

Como se puede comprobar, el código XHTML es muy similar al equivalente en HTML. Básicamente, la diferencia está en la declaración del tipo de documento: en el primer caso se indica que es un documento **xml**, mientras que en el segundo se trata de **html**. Además,

© Ediciones Paraninfo

en el documento XHTML se especifica el espacio de nombres mediante el atributo **xmlns** y el idioma mediante el atributo **lang.**

No existen más diferencias y el resultado visual generado por el navegador en ambos casos es exactamente el mismo (Figura 2.75).

Figura 2.75. *Los navegadores más utilizados saben procesar XHTML.*

Las diferencias entre HTML y XHTML comienzan a ser evidentes cuando el documento no cumple con las reglas del lenguaje: sustituyendo el elemento **<body>** de los ejemplos anteriores con el siguiente código, el documento HTML se verá correctamente mientras que el documento XHTML mostrará un error por no estar cerrado el elemento **
** y por abrir el elemento **<p>** con letras minúsculas y cerrarlo con letras mayúsculas.

A continuación, se muestra el código del elemento <body> con las modificaciones que lo convierten en un documento XML **mal** formado:

```
<body>
    <h3>¡HOLA!</h3>
    <br>
    <p>El día comenzó alegre y luminoso</P>
</body>
```

Figura 2.76. *Si un documento XHTML no cumple con las reglas de XML, el navegador lo detecta.*

© Ediciones Paraninfo

2.8. Herramientas de consulta y edición de HTML

La selección de una herramienta para trabajar con HTML o XHTML depende del uso que se vaya a dar. Existen dos tipos de herramientas: las herramientas de consulta y las herramientas de edición.

Las herramientas de consulta se utilizan para visualizar el resultado tras procesar el documento HTML. Por otra parte, las herramientas de edición son las que permiten crear o modificar documentos HTML. En algunos casos, las herramientas son válidas para ambas operaciones.

Una de las características de los lenguajes de marcas es que están escritos en texto plano, lo que facilita enormemente su procesamiento. Cualquier editor, por humilde que sea, sirve para escribir una página web en HTML. La visualización, en cambio, es más compleja, ya que el estándar de HTML es extenso y hay que implementarlo de manera completa.

Sin ser exhaustivos, en esta unidad se muestran algunas de las herramientas para editar o visualizar documentos HTML desde un ordenador.

2.8.1. Navegadores web

La web sin los navegadores no tendría razón de ser. Son los programas que permiten conectar un ordenador o cualquier otro dispositivo con un servidor, recuperar el código HTML, interpretarlo y convertirlo en una representación visual de acuerdo con el estándar del W3C.

Son, principalmente, programas de consulta, pero aportan más funcionalidad relacionada con el código HTML.

Los navegadores más utilizados son los siguientes:

- Chrome, de Google.
- Opera, de Opera Software.
- Safari, de Apple.
- Firefox, de la Fundación Mozilla.
- Edge, de Microsoft.
- Explorer, de Microsoft.

Existen muchos más navegadores, con menores cuotas de mercado, pero también interesantes; Torch, SeaMonkey, Lynx (solo dispone de modo texto) o Vivaldi son algunos ejemplos.

Además de mostrar el resultado de interpretar la página web, los navegadores permiten consultar el código fuente pulsando en el botón derecho del ratón sobre la página y seleccionando la opción «Ver código fuente de la página» o la combinación de teclas Control + U.

© Ediciones Paraninfo

Figura 2.77. *Desde el navegador se puede consultar el código HTML origen de la página.*

Nota técnica

La opción «Ver código fuente de la página» aparece en los navegadores Chrome, Firefox y Microsoft Edge. En el navegador Opera la opción equivalente se llama «Origen de la página», mientras que en Safari la opción es «Mostrar código fuente de la página». Dependiendo de las versiones de los navegadores y del idioma de estos, las opciones pueden llamarse de otra forma. En cualquier caso, el resultado es el mismo: el navegador muestra el código HTML descargado del servidor.

Además de la posibilidad de consultar el código fuente, la mayoría de los navegadores modernos disponen de una serie de herramientas conocidas como **herramientas para desarrolladores** (también denominadas informalmente ***devtools***, en inglés).

El acceso a estas herramientas se realiza de forma distinta en función del navegador. Por ejemplo, en Google Chrome se hace a través de la opción «Más Herramientas / Herramientas para desarrolladores», mediante la combinación de teclas Control + Mayúsculas + I o pulsando la tecla F12.

Entre las herramientas para desarrolladores, tomando Google Chrome como referencia, se encuentran las siguientes pestañas relativas a HTML:

- *Elements.* Permite consultar la estructura del documento HTML y el estilo CSS aplicado a cada elemento. Se puede modificar los elementos HTML y los estilos y ver en tiempo real el resultado de las modificaciones.
- *Sources.* Se muestran todos los recursos descargados relativos a la página HTML: el propio documento, imágenes, hojas de estilo CSS, ficheros JavaScript externos, etc. Permite consultar el código HTML.

© Ediciones Paraninfo

2.8.2. Editores de texto

La edición de ficheros HTML puede realizarse desde cualquier editor por básico que sea, ya que están escritos en texto plano.

Programas como Notepad de Microsoft Windows o TextEdit de los sistemas macOS en los ordenadores Apple son suficientes para editar ficheros HTML.

Estos editores simples permiten crear y modificar los ficheros, pero no proporcionan ninguna ayuda, por lo que es recomendable utilizar otros editores más sofisticados como Visual Studio Code, Atom, Brackets, Sublime Text, UltraEdit, Notepad++ o algún otro editor similar. Todos ellos integran herramientas de corrección de sintaxis y de sugerencias que facilitan la creación de documentos. También suelen disponer de esquemas de color que ayudan en la lectura del código.

Figura 2.78. *Los esquemas de color de los editores ayudan a diferenciar elementos, atributos y textos.*

Complemento Emmet

Merece la pena hacer referencia en este punto al complemento *Emmet,* disponible en la mayoría de los editores modernos y que proporciona una serie de atajos para la creación automática de código.

Por ejemplo, con el *plugin Emmet* instalado, la creación de una lista ordenada que contenga cinco ítems de la clase «reserva» se reduce a escribir:

```
ol>li.reserva*3
```

Generando el siguiente código HTML:

```
<ol>
    <li class="reserva"></li>
    <li class="reserva"></li>
    <li class="reserva"></li>
</ol>
```

© Ediciones Paraninfo

Actividad resuelta 2.5

Creación de documento HTML con *Emmet*

Crear una página HTML utilizando la ayuda del complemento *Emmet* con el programa Visual Studio Code.

Solución

- Crear un fichero con el nombre index.html.
- Escribir en la primera línea **html:5** y pulsar el tabulador. Aparecerá la estructura de una página HTML, incluyendo las etiquetas principales, el elemento **<head>** y el elemento **<body>**.
- Dentro del elemento **<body>** escribir **div#principal** y pulsar el tabulador. Se creará un bloque ** ** con el identificador **principal.**
- Dentro del elemento ** ** que se acaba de crear, escribir **table>tr*3>td.centrado*2.** Se creará una tabla con tres filas y dos columnas, en las que cada celda pertenecerá a la clase «**centrado**».

2.8.3. Editores visuales de HTML

Un nivel por encima de los editores de texto se encuentran los editores visuales, también denominados WYSIWYG (acrónimo de *What You See Is What You Get* o «lo que ves es lo que tienes»).

Estos editores permiten construir documentos HTML con herramientas gráficas y simplifican el proceso de creación hasta el punto de que no obligan a conocer el lenguaje en profundidad, aunque esto es recomendable porque a menudo es necesario modificar el código HTML generado automáticamente. Su uso estuvo muy generalizado, pero las nuevas técnicas de diseño y programación de aplicaciones web les han hecho perder importancia.

Los mejores editores están desarrollados con software privativo y su licencia tiene un coste económico, pero existen alternativas gratuitas de gran calidad.

Algunas de estas herramientas son (aunque existen muchas más) Adobe Dreamweaver (comercial), BlueGriffon (gratuito con algunos componentes comerciales) y HTML-Notepad (gratuito).

2.8.4. Editores online

Editar un documento HTML es una tarea que puede resultar compleja y llevar mucho tiempo. Durante la escritura de la página es frecuente que el autor pruebe diferentes alternativas hasta encontrar aquella que más le convence. Esto supone escribir el fichero con un editor, guardarlo, abrirlo con el navegador web, modificarlo, guardarlo de nuevo, refrescar la página en el navegador y repetir estos pasos hasta finalizar.

Este proceso de modificación y visualización, especialmente válido para la realización de prototipos o de pruebas de concepto, se puede realizar de manera cómoda utilizando algunos servicios gratuitos disponibles en la red, como JSFiddle, CodePen o Liveweave.

© Ediciones Paraninfo

Estos sitios web ofrecen editores online que permiten introducir código HTML, CSS y JavaScript, y comprobar en la misma página y en tiempo real el resultado.

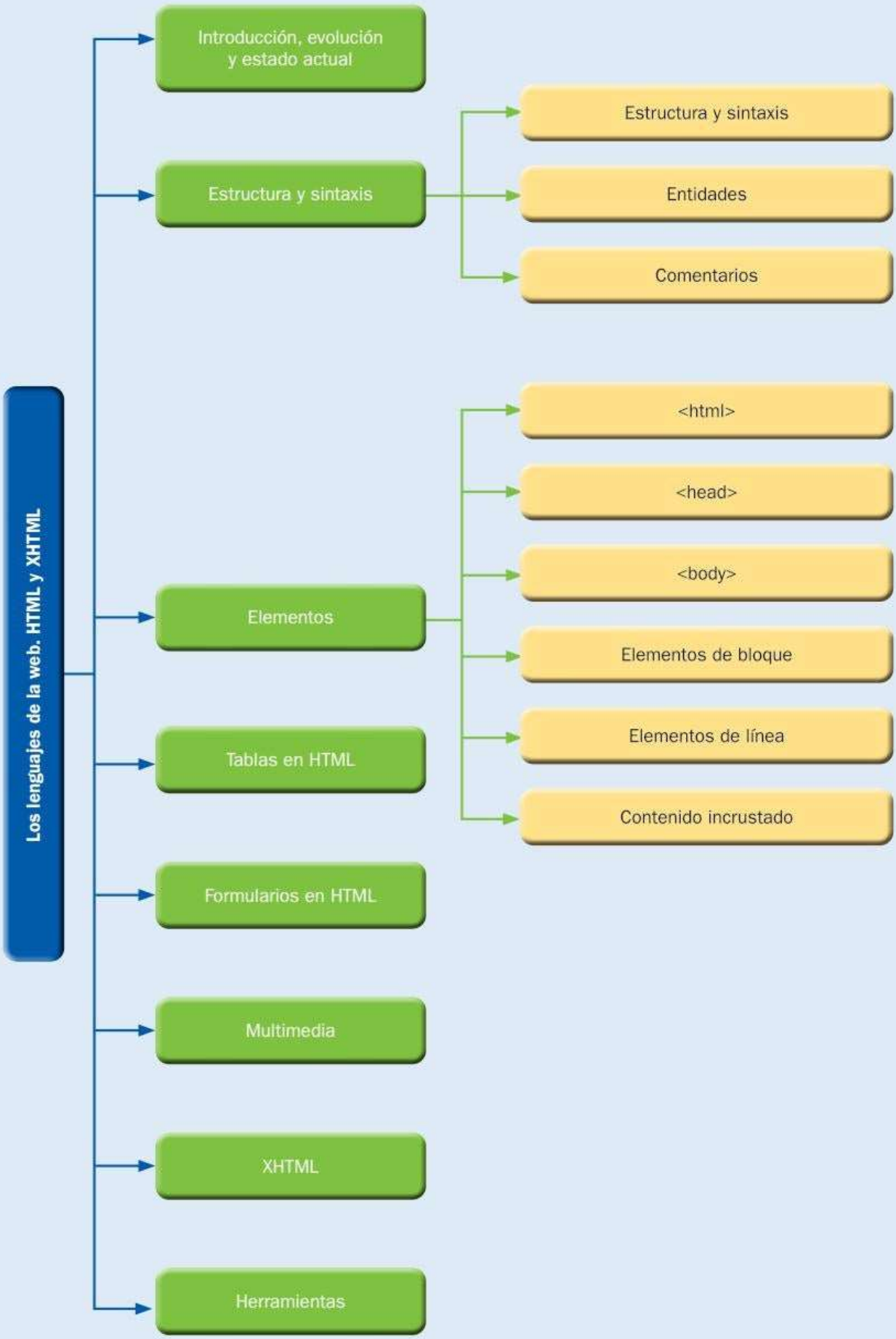
En la Figura 2.79 se muestra un ejemplo de uso de JSFiddle, en la que se puede observar el código HTML junto con su representación.

Figura 2.79. *Con los editores online se puede editar el código HTML y ver el resultado simultáneamente.*

Los editores en línea son muy útiles para hacer pruebas por la agilidad que proporcionan. No obstante, es recomendable disponer de al menos un editor instalado en el ordenador. Los editores de escritorio son más potentes y disponen de características que no proporcionan los editores en línea.

© Ediciones Paraninfo

MAPA CONCEPTUAL



© Ediciones Paraninfo

Actividades de comprobación

2.1. ¿Quién diseñó el lenguaje de marcas HTML?

a) Bill Gates.
b) Tim Berners-Lee.
c) Steve Jobs.
d) Jeff Bezos.

2.2. ¿Qué significan las siglas HTML?

a) *HiperTerminal Markup Language.*
b) *HyperText Master Language.*
c) *HyperText Markup Language.*
d) *HyperText Markup Line.*

2.3. ¿Cuál de las siguientes etiquetas se usa para marcar el cuerpo de un documento HTML?

a) <head>
b) <folder>
c) <body>
d)

2.4. ¿Cuál de las siguientes entidades representa un espacio en blanco?

a) ≮
b) '
c)
d) "

2.5. ¿Cómo se marcan los comentarios en HTML?

a) <"-- COMENTARIO -->
b) <?-- COMENTARIO -->
c) </-- COMENTARIO -->
d) <!-- COMENTARIO -->

2.6. ¿Qué atributo oculta un elemento HTML y hace que no se represente en el navegador?

a) contenteditable.
b) id.
c) hidden.
d) spellcheck.

2.7. ¿Dónde se muestra el valor del elemento <title>?

a) En la primera línea de la página.
b) En la URL de la página.
c) En la pestaña del navegador.
d) En la barra de direcciones.

2.8. El elemento <article>:

a) Suele contener el menú de navegación.
b) Agrupa información concebida como una unidad independiente.
c) Suele agrupar información relativa a los derechos de autor y propiedad intelectual.
d) Muestra el contenido dentro de una caja de diálogo.

© Ediciones Paraninfo

2.9. El elemento <dialog>:

a) Suele contener el menú de navegación.
b) Agrupa información concebida como una unidad independiente.
c) Suele agrupar información relativa a los derechos de autor y propiedad intelectual.
d) Muestra el contenido dentro de una caja de diálogo.

2.10. Las listas de definición se construyen con los elementos:

a) <dl>, <dt> y <dd>.
b) <ol> y <li>.
c) <ul> y <li>.
d) <table>, <tr> y <td>.

2.11. Las listas ordenadas se construyen con los elementos:

a) <dl>, <dt> y <dd>.
b) <ol> y <li>.
c) <ul> y <li>.
d) <table>, <tr> y <td>.

2.12. El elemento permite:

a) Crear divisiones y agrupar contenidos.
b) Dividir las listas de definición en apartados.
c) Mostrar fórmulas matemáticas.
d) Dividir el contenido de la página en datos y estilo.

2.13. El elemento utilizado para crear formularios es:

a)
b)
c) <form>
d) <aside>

2.14. El atributo del elemento <a> que contiene la URL del hiperenlace es:

a) target.
b) source.
c) href.
d) src.

2.15. Indica cuál de los siguientes elementos es un componente incrustado:

a) <input>
b) <img>
c) <a>
d) <iframe>

2.16. ¿Qué elemento de HTML marca el texto con un color llamativo como un rotulador?

a) <mark>
b)
c) <code>
d) <b>

© Ediciones Paraninfo

2.17. ¿Qué valor del atributo type del elemento <input> hace que solo admita valores numéricos?

a) text.
b) date.
c) number.
d) int.

2.18. El elemento que muestra su contenido textual tachado e indica que no está vigente es:

a) <obsolete>
b) <s>
c) <del>
d) <mark>

2.19. El elemento :

a) Es un elemento de línea.
b) Permite asignar un estilo a un fragmento de contenido.
c) No introduce saltos de línea.
d) Todas las respuestas son correctas.

2.20. El elemento para mostrar un subíndice es:

a) <i>.
b) <sub>.
c) <sup>.
d) Todas las respuestas son correctas.

2.21. El elemento de HTML que declara una tabla es:

a)
b) <table>
c)
d) <thead>

2.22. El elemento de HTML que crea una fila de datos en una tabla es:

a) <tr>
b) <td>
c) <th>
d)

2.23. El atributo action de un formulario de HTML permite:

a) Elegir el método de envío de los datos del formulario entre GET y POST.
b) Definir el destino de los datos.
c) Definir una acción que se ejecutará cuando se pulse un botón cualquiera.
d) Definir el nombre del formulario al que pertenece.

2.24. El indicador del grado de avance del elemento <progress> se mueve de derecha a izquierda y de izquierda a derecha:

a) Siempre.
b) Nunca.
c) Cuando se le asigna valor al atributo value.
d) Cuando no se le asigna valor al atributo value.

© Ediciones Paraninfo

2.25. ¿Qué atributo hay que incluir en un campo <input> para que sea obligatorio?

a) min.
b) max.
c) readonly.
d) required.

2.26. HTML permite incluir en una página:

a) Ficheros de sonido.
b) Imágenes.
c) Ficheros de vídeo.
d) Todas las respuestas son correctas.

2.27. Un documento XHTML es:

a) Un documento HTML.
b) Un documento XML.
c) Un documento SVG.
d) Un documento CSS.

Actividades de aplicación

2.28. Enumera las versiones por las que ha pasado el lenguaje HTML.

2.29. Describe la estructura básica de un documento HTML.

2.30. Explica cuál es la diferencia entre los elementos en bloque y los elementos en línea.

2.31. Crea un documento HTML en el que se incluya una cita de algún personaje histórico. Utiliza el elemento <blockquote>.

2.32. Emplea en una página HTML los elementos <details> y <sumary>.

2.33. Usa en una página HTML el elemento <dialog>.

2.34. Crea una lista de definición en HTML explicando los géneros cinematográficos siguientes: «Acción», «Terror» y «Comedia».

2.35. En una página HTML incluye una imagen con el elemento <figure>. Utiliza el elemento <figcaption> para agregar una leyenda asociada a dicha imagen.

2.36. Crea el cuerpo de una página HTML con la siguiente estructura: «Capítulo 1», dividido en las secciones «Introducción» y «Objetivos»; «Capítulo 2», dividido en las secciones «Formatos», «Estándares» y «Aplicaciones». En cada sección agrega dos párrafos que contengan cualquier texto. Utiliza los elementos <h1>, <h2> y <p>.

2.37. Crea una lista ordenada en HTML con los signos del zodíaco. Haz que cada elemento sea un hiperenlace que apunte a la página de Wikipedia correspondiente.

2.38. En una página HTML haz uso de los elementos <ins> y <del>.

© Ediciones Paraninfo

2.39. En una página HTML emplea los elementos <sub> y <sup>.

2.40. Crea un documento HTML que contenga el tráiler de una película. Utiliza el código <iframe> proporcionado por el portal YouTube.

2.41. Realiza una tabla en HTML en la que se muestren los ingresos y gastos de cada día de la semana de una empresa. Inventa los datos.

2.42. Crea un formulario HTML de suscripción a una revista. Debe solicitar los siguientes datos: nombre y apellidos (alfanumérico, 100 caracteres máximo, obligatorio), dirección postal (alfanumérico, 100 caracteres máximo, obligatorio), número de ejemplares (numérico, valor mínimo 1 y máximo 10) y tipo de suscripción (a elegir entre mensual, trimestral y anual). Deberá disponer de un botón de envío y otro de reinicio.

2.43. Crea una página web que permita reproducir un vídeo alojado en internet. Utiliza el elemento <video>.

Actividades de ampliación

2.44. Busca información relativa al primer servidor web que usó Tim Berners-Lee. Compara las características de dicho servidor con las de un ordenador moderno.

2.45. Averigua cuál es el portal web de la comunidad autónoma en la que resides. Una vez en él, accede al código fuente, localiza las etiquetas <meta> e investiga para qué sirven aquellas que no conozcas.

2.46. Busca un ejemplo de tabla de datos en Wikipedia. En las páginas de los deportistas, por ejemplo, suele haberlas. Consulta el código fuente y observa la estructura con detenimiento. Construye una tabla reproduciendo dicha estructura.

2.47. Profundiza en el uso de las expresiones regulares como herramienta de validación de campos de entrada de texto.

2.48. Halla más información sobre el uso de los elementos multimedia en HTML 5.

2.49. Crea un documento XHTML con diversos errores. Abre el fichero con un navegador y estudia los mensajes de error. Soluciónalos de uno en uno.

2.50. Profundiza en el conocimiento del complemento *Emmet* en relación con la construcción de tablas en HTML. Haz pruebas en el editor Visual Studio Code, que ya tiene *Emmet* instalado por defecto.

© Ediciones Paraninfo

Enlaces web de interés

- **HTML w3c.es** - https://html.spec.whatwg.org/multipage/
  (Información oficial sobre HTML)
- **MDN Web Docs - HTML** - https://developer.mozilla.org/es/docs/Web/HTML
  (Web de Mozilla sobre HTML)
- **w3schools** - https://www.w3schools.com/html/
  (Tutorial de HTML de w3schools)
- **Visual Studio Code** - https://code.visualstudio.com/
  (Sitio web del editor Visual Studio)
- **Atom** - https://atom.io/
  (Sitio web del editor Atom)
- **Brackets** - http://brackets.io/
  (Sitio web del editor Brackets)
- **Sublime Text** - https://www.sublimetext.com/
  (Sitio web del editor Sublime Text)
- **UltraEdit** - https://www.ultraedit.com/
  (Sitio web del editor UltraEdit)
- **Notepad++** - https://notepad-plus-plus.org/
  (Sitio web del editor Notepad++)
- **Emmet** - https://emmet.io/
  (Sitio web del componente Emmet)
- **JSFiddle** - https://jsfiddle.net/
  (Editor online JSFiddle)
- **CodePen** - https://codepen.io/
  (Editor online CodePen)
- **Liveweave** - https://liveweave.com/
  (Editor online Liveweave)

© Ediciones Paraninfo

UNIDAD 3

Definición de estilos en entornos web: CSS

Objetivos

- Conocer la estructura y la sintaxis de CSS.
- Aprender cómo se aplica CSS a los documentos.
- Presentar los tipos utilizados para determinar colores, tipos de letras y unidades de medida.
- Enumerar y descubrir los diferentes tipos de selectores.
- Desglosar las propiedades que se pueden modificar desde los estilos CSS.

Contenidos

- 3.1. Introducción, evolución y estado actual de CSS
- 3.2. Estructura y sintaxis de CSS
- 3.3. Aplicación de CSS
- 3.4. Prioridades y orden en CSS
- 3.5. Valores y unidades en CSS
- 3.6. Selectores CSS
- 3.7. Propiedades CSS

Introducción

CSS *(Cascading Style Sheets)* es un potente mecanismo para agregar estilo (colores, tipos de letra, espaciados, etc.) a las páginas web, pero no es un lenguaje de marcas. La pregunta que surge a continuación es obvia: ¿por qué tratar CSS en este libro?

Con el paso del tiempo y tras varias versiones, HTML ha dejado de contener información referente a cómo se deben presentar los datos en las páginas web, delegándose esta tarea en las hojas de estilo CSS. Solo con HTML las presentaciones de las páginas serían extremadamente austeras y carentes de atractivo. CSS está presente en casi todos los documentos HTML existentes, por lo que no se concibe una tecnología sin la otra. Por esta razón es necesario incluir CSS en este libro, ya que se puede considerar que forma con HTML un equipo inseparable.

3.1. Introducción, evolución y estado actual de CSS

CSS son las siglas de *Cascading Style Sheets* («hojas de estilo en cascada»), y es el lenguaje que se utiliza para definir el aspecto de las páginas HTML y XHTML.

Se diseñó para separar los datos (contenidos en los documentos HTML) de las reglas de presentación (contenidos en los documentos CSS), ya que, con anterioridad a su aparición, las reglas de presentación formaban parte de las etiquetas HTML.

Esta separación proporciona una enorme versatilidad a la hora de presentar la información, pues un mismo documento HTML puede combinarse con diferentes hojas de estilo CSS para generar distintas presentaciones.

En las figuras 3.1 y 3.2 se muestra el mismo documento HTML utilizando distintas hojas de estilo CSS. Las diferencias son evidentes: la tipografía, el color del texto, el color del fondo, la presencia o ausencia de las viñetas de la lista y la alineación del texto. Estos cambios se han introducido utilizando distintas hojas CSS sobre un mismo contenido HTML.

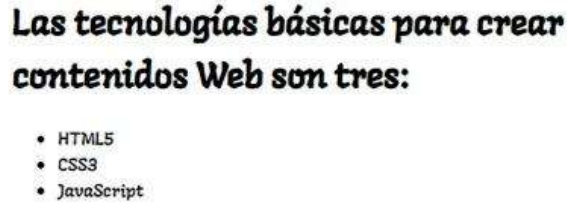


Figura 3.1. *Con CSS se asigna estilo a los documentos HTML.*

Figura 3.2. *Un mismo documento HTML puede tener múltiples presentaciones.*

Al igual que HTML, es el consorcio W3C la entidad encargada del mantenimiento y la estandarización de CSS. La primera versión CSS vio la luz en 1996 y desde entonces ha habido sucesivas modificaciones hasta llegar a la versión actual, denominada CSS3. No obstante, hay que aclarar algunos aspectos relacionados con las versiones. Hasta la versión 2.1, todas las anteriores se presentaban como una especificación única. Por cuestio-

© Ediciones Paraninfo

nes de organización, el W3C decidió que la especificación de la versión 3 se realizaría por módulos, por lo que ya no existe una única versión actual, sino módulos actualizados a la versión 3 y módulos en versión 2.1. Sea como sea, en la jerga técnica se hace referencia a CSS3 como la versión actual de CSS, sin entrar en más detalles.

En la Tabla 3.1 se muestran las diferentes versiones y su año de publicación.

Tabla 3.1. Evolución de las versiones de CSS

AÑO	VERSIÓN	OBSERVACIONES
1996	CSS1	El W3C ya no mantiene esta versión.
1998	CSS2	El W3C ya no mantiene esta versión.
2011	CSS2.1	Corrige errores de CSS2.
2012	CSS3	Esta versión está dividida en módulos, por lo que no existe una única fecha global de cambio de versión.

Nota técnica

Todas las referencias realizadas en esta unidad a HTML afectan de manera implícita a XHTML, ya que el modelo de trabajo en relación con CSS es idéntico en ambos lenguajes.

3.2. Estructura y sintaxis de CSS

El funcionamiento de CSS consiste en definir unas reglas de presentación que se van a aplicar a un número indeterminado de elementos del documento HTML, al que están vinculadas. Por lo tanto, se necesitan dos herramientas básicas para crear y aplicar un estilo:

- **Selectores:** son las herramientas que permiten seleccionar el elemento o elementos sobre los que aplicar las reglas.
- **Declaraciones:** son las indicaciones para asignar, mediante pares de propiedad-valor, el aspecto deseado a los elementos determinados por el selector.

Los selectores y las declaraciones se agrupan en **reglas.**

En el siguiente ejemplo se muestra una regla compuesta por los distintos elementos:

- Selector **body.** El contenido se aplica a todos los elementos que se encuentren dentro del elemento **<body>** en la página HTML.
- Las declaraciones que asignan valor a las propiedades **text-align, font-family, background-color** y **color,** que afectarán respectivamente a la alineación del texto, el tipo de letra, el color de fondo y el color del elemento **<body>** y de todos los elementos contenidos en este, siempre que las propiedades sean aplicables a dichos elementos.

© Ediciones Paraninfo

```
body {
        text-align: center;
        font-family: 'Roboto', sans-serif;
        background-color: black;
        color: white;
}
```

Los comentarios se delimitan con los símbolos /* y */ y no se procesan por el navegador.

```
/* Este texto es un comentario */
```

3.3. Aplicación de CSS

Para que las reglas CSS tengan efecto sobre los elementos de un documento HTML deben vincularse con este. Esto se puede realizar de tres maneras distintas:

- **Como un documento CSS externo.** Las reglas se almacenan en un documento externo con extensión .css y se establece una relación entre el documento HTML y dicho fichero. La asignación se hace mediante el elemento **<link>**, asignando «stysheet» como valor al atributo **rel** y la URI del fichero .css al atributo **href.**

Ejemplo:

```
<!DOCTYPE html>
<html lang="es">
<head>
   <meta charset="UTF-8">
   <link rel="stylesheet" href="ejemplo1_dark.css">
   <title>Título del documento</title>
</head>
<body>
   <H1>Las tecnologías básicas para crear contenidos Web son tres:</
H1>
   <ul>
      <li>HTML5</li>
      <li>CSS3</li>
      <li>JavaScript</li>
   </ul>
</body>
</html>
```

ejemplo1_dark.css

```
body {
   text-align: center;
```


© Ediciones Paraninfo

```
        background-color: black;
        color: white;
    }
    ul {
        list-style: none;
    }
```

- **Como una declaración CSS interna.** Las reglas se declaran en un elemento **<style>** incrustado en el propio documento HTML.

Ejemplo:

```
<!DOCTYPE html>
<html lang="es">
<head>
    <style>
    body {
        text-align: center;
        background-color: black;
        color: white;
    }
    ul {
        list-style: none;
    }
    </style>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <title>Document</title>
</head>
<body>
    <H1>Las tecnologías básicas para crear contenidos Web son tres:</
H1>
    <ul>
        <li>HTML5</li>
        <li>CSS3</li>
        <li>JavaScript</li>
    </ul>
</body>
</html>
```

- **Como una asignación *inline*** (en el propio elemento) de las reglas. Las reglas se declaran en el valor de la propiedad **style** del elemento al que afectan.

Ejemplo:

```
<!DOCTYPE html>
<html lang="es">
```

© Ediciones Paraninfo

```
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-
scale=1.0">
   <title>Document</title>
</head>
<body style="text-align: center;background-color: black;color:white">
   <H1>Las tecnologías básicas para crear contenidos Web son tres:</
H1>
   <ul style="list-style: none;">
      <li>HTML5</li>
      <li>CSS3</li>
      <li>JavaScript</li>
   </ul>
</body>
</html>
```

Las tres alternativas no son exclusivas; se puede añadir una referencia a una o más hojas de estilo externas, incluir un elemento **<style>** y añadir un atributo **style** a un elemento. En ningún caso estas alternativas son incompatibles.

Cada una de las opciones para agregar estilos tiene sus ventajas e inconvenientes, siendo la más versátil y reutilizable la solución de crear un documento externo, ya que se podrá utilizar desde diferentes documentos. No obstante, las otras opciones a veces son elegidas por comodidad o para realizar prototipos rápidos, teniendo siempre en cuenta que las reglas no podrán reutilizarse con comodidad.

Actividad propuesta 3.1

Aplicación de estilo

Escribe en un editor los siguientes documentos HTML y CSS. Vincula el documento CSS con el documento HTML. A continuación abre el documento HTML con un navegador y observa el resultado.

Código HTML:

```
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <title>Aplicando estilos</title>
    <!--AGREGAR EL VÍNCULO CON LA HOJA DE ESTILOS CSS-->
</head>
<body>
    <h1>Leonardo Da Vinci</h1>
    <p>El gran genio del <span class="ciudad">Renacimiento</span> italiano nació
en Anchiamo en el año 1452</p>
</body>
</html>
```


© Ediciones Paraninfo

Código CSS:

```
h1 {
    text-transform: uppercase;
}
p {
    font-size: 1.5em;
    color: #888888;
}
.ciudad {
    font-style: italic;
    font-weight: bolder;
}
```

3.4. Prioridades y orden en CSS

El número de reglas que se puede aplicar a un documento HTML es ilimitado y estas pueden entrar en conflicto. Pendientes de enumerar y explicar los diferentes tipos de selectores, se puede adelantar que dos reglas, una que utiliza el selector **body** y otra que utiliza el selector **h1**, afectarán a los elementos **<h1>**. La pregunta que surge es ¿qué regla va a elegir el navegador en caso de que dos o más apliquen al mismo elemento? La respuesta está en la «C» de CSS.

Antes de explicar cómo se asignan las prioridades, hay que saber que pueden existir tres hojas de estilo distintas en una página: la propia del navegador, que es la primera que se aplica por defecto; la que proporciona el usuario, que se asocia al navegador, se aplica por defecto a todas las páginas y es muy útil para personas con discapacidad, y por último, están las hojas de estilo del diseñador, que se aplican al final y pueden ser más de una.

Hay tres conceptos fundamentales en la aplicación de los estilos: **cascada, especificidad** y **herencia.**

La **herencia** provoca que los valores de algunas de las propiedades aplicadas a un elemento contenedor se apliquen también a los elementos contenidos por él. No todas las propiedades se ven afectadas por este comportamiento, ya que en algunos casos no tendría sentido.

Los estilos CSS se aplican en **cascada,** según unas reglas fijas, obteniendo unos resultados predecibles. Los factores que participan en la decisión de qué estilo aplicar son los siguientes:

- **Origen e importancia:** los estilos inherentes al navegador tienen menos prioridad que los elegidos por el usuario y estos, a su vez, menos que los creados por el diseñador de la página, que son los que se acaban imponiendo en caso de conflicto. Dentro de los estilos proporcionados por el diseñador, influye si están definidos *inline* (mayor prioridad), en un elemento **<style>** (prioridad intermedia) o en un recurso externo (menor prioridad).
- **Nivel de especificidad:** cuanto más específico, mayor prioridad.

© Ediciones Paraninfo

- **Orden de aparición:** los últimos estilos en leerse o procesarse se imponen a los existentes.

De manera más detallada, las reglas fundamentales de aplicación de estilos en caso de conflicto son las siguientes:

- La regla definida con el selector universal (*) se impone al resto de reglas definidas con selectores más específicos.
- Si una propiedad se modifica en un único punto, no hay conflicto y se aplica.
- Si una propiedad se modifica en el atributo **style**, se impone al intento de modificación de la misma propiedad en el elemento **<style>** o en una hoja CSS externa.
- Si una propiedad se modifica en el elemento **<style>**, se impone al intento de modificación de la misma propiedad en una hoja CSS externa.
- Si dos o más hojas CSS modifican la misma propiedad, se impone la hoja que se ha añadido en último lugar.
- Si dos reglas modifican la misma propiedad del mismo elemento, se impone aquella cuyo selector es más específico (a excepción del selector universal):
  - El selector con mayor número de **identificadores** (**id**) será el de mayor prioridad.
  - El selector con mayor número de **clases** (**class**) o pseudoclases será el siguiente en el orden de prioridad.
  - El selector con mayor número de elementos o pseudoelementos será el siguiente en el orden de prioridad.
- En caso de igualdad en las prioridades se impone el último estilo aplicado.

Estas reglas generales se pueden omitir o modificar mediante una serie de valores comunes a todas las propiedades que tienen efecto sobre cómo se realiza la herencia. Estos valores son:

- **inherit.** Activa la herencia, haciendo que la propiedad en la que se está asignando como valor herede la configuración del elemento padre.
- **initial.** Al indicar este valor a una propiedad se consigue que tome el valor de la hoja de estilos por defecto para el elemento al que hace referencia.
- **unset.** Restablece el valor de la propiedad a su valor natural.
- **revert.** Restablece el valor de la propiedad al valor que habría tenido si no hubiese sufrido cambios desde ningún origen.

Además, CSS dispone de la declaración **!important.** Este atributo anula las propiedades de un elemento como consecuencia de la aplicación de las reglas de asignación.

La declaración se efectúa inmediatamente después de la asignación del valor a la propiedad y junto antes del carácter ; que finaliza la asignación. En el siguiente código CSS se puede observar un ejemplo de aplicación:

```
div {
    color:rgb(9, 145, 224) !important;
}
```


© Ediciones Paraninfo

Actividad propuesta 3.2

Prioridades

Dado el siguiente bloque de código HTML y estilo CSS, determina de qué color aparecerán estas cuatro citas del poeta italiano Dante Alighieri.

```
<p class="frases-celebres" id="frase1">
Aquél que escucha bien, toma apuntes.
</p>
<p id="frase2">
Si no se modera tu orgullo, él será tu mayor castigo
</p>
<p class="frases-celebres">
Una poderosa llama sigue a una pequeña chispa.
</p>
<p>
El secreto para que las cosas sean hechas está en hacerlas.
</p>

#frase1 {
    color: blue;
}

.frases-celebres {
    color: green;
}

p {
    color: red;
}
```

3.5. Valores y unidades en CSS

Antes de presentar las posibilidades de las hojas de estilo es necesario conocer qué valores admiten los diferentes parámetros y en qué unidades se expresan.

3.5.1. Tipos de datos

Los tipos de datos que se utilizan a la hora de configurar un estilo son los que se detallan en la Tabla 3.2.

Tabla 3.2. Tipos de datos admitidos por CSS

Tipo	Descripción
Entero	Número entero, ya sea positivo o negativo. Por ejemplo 10 o –3.
Número	Un número decimal. Utiliza el punto como separador decimal. Si no hay decimales, se omite el punto.

© Ediciones Paraninfo

Tipo	Descripción
Dimensión	Es un número con una dimensión asociada; deg (grados), s (segundos) o px (píxeles) son algunos ejemplos.
Porcentaje	Una fracción de un total, normalmente referente a la dimensión del padre del elemento al que se está aplicando. Por ejemplo 60 % en un **font-size** representa dicho porcentaje del tamaño del texto del elemento padre. Si, en cambio, se aplica un porcentaje a la propiedad **width,** se aplica dicho porcentaje sobre el ancho del elemento contenedor.

3.5.2. Unidades de longitud absoluta

Las unidades de longitud absoluta representan longitudes sin importar en qué contexto se encuentre el componente al que hacen referencia.

Tabla 3.3. Unidades absolutas

Unidad	Nombre
px	Píxel
cm	Centímetro (96 px / 2,54)
mm	Milímetro
Q	Cuarto de milímetro
in	Pulgada (96 px)
pt	Puntos (1/72 de in)
pc	Picas (1/16 de in)

3.5.3. Unidades de longitud relativas

Las unidades de longitud relativas representan longitudes que tienen diferente valor en función del contexto en el que se encuentran.

Tabla 3.4. Unidades relativas

Unidad	Nombre
em	Tamaño de letra del elemento padre.
ex	Altura de la fuente del elemento.
ch	Ancho del carácter «0» de la fuente del elemento.
rem	Tamaño de letra del elemento raíz.

© Ediciones Paraninfo

Unidad	Nombre
lh	Altura de la línea del elemento.
vw	1 % del ancho de la ventana gráfica.
vh	1 % del alto de la ventana gráfica.
vmin	1 % de la dimensión más pequeña de la ventana gráfica.
vmax	1 % de la dimensión más grande de la ventana gráfica.

De este conjunto de posibles unidades, las más utilizadas son **em** y **rem.** Las dimensiones absolutas son más sencillas de aplicar, pero los resultados dependen mucho de la resolución y de la densidad de píxeles de la pantalla, por lo que suele ser más eficaz utilizar unidades relativas.

Ejemplo:

Se muestran tres textos con diferentes tamaños de letra: en los dos primeros casos se utiliza **em** como unidad de medida mientras que en el tercer caso se asigna un tamaño porcentual. El tamaño asignado al id **saludoTriplePorcentual** (300 %) y al id **saludoTriple** (3 em) tienen el mismo efecto, ya que ambos multiplican por tres el tamaño del tipo de letra del elemento.

```
<div>
   Buenos días normal
   <div id="saludoTriplePorcentual">
      Buenos días porcentual
   </div>
</div>
<div id="saludoDoble">
     Buenos días doble
</div>
<div id="saludoTriple">
     Buenos días triple
</div>
```

```
#saludoDoble {
   font-size: 2em;
}
#saludoTriple {
  font-size: 3em;
}
#saludoTriplePorcentual {
   font-size: 300%;
}
```

© Ediciones Paraninfo

Buenos días normal

Buenos días porcentual

Buenos días doble

Buenos días triple

Figura 3.3. *En CSS se pueden expresar los tamaños en varios tipos de unidades.*

3.5.4. Colores

Los colores se pueden utilizar para colorear textos, fondos de contenedores o bordes, por ejemplo. CSS permite expresar los colores con varias notaciones distintas:

- Por nombre. Indicando el nombre del color de la lista de la recomendación del W3C *(CSS Color Module Level 3)*. Así, el color rojo se asigna con el valor **red.** La lista completa se encuentra en el siguiente enlace: https://www.w3.org/TR/css-color-3/#html4.
- Por valor RGB (del inglés *Red, Green, Blue;* «rojo, verde, azul»). Indicando la composición de las tres componentes con valores en el rango 0-255 mediante el formato **rgb(*r,g,b*)**. Por ejemplo el color rojo se asigna con el valor **rgb(255,0,0).** Con la variante **rgba(*r,g,b,a*)** se añade un componente que indica la transparencia del color (lo que se denomina **canal alfa**) con un valor decimal comprendido entre 0 (completamente transparente) y 1 (completamente opaco).
- Por valor hexadecimal. Indicando el valor en hexadecimal de las componentes RGB mediante el formato ***#valorhexadecimal.*** Por ejemplo, el color rojo se asigna con el valor **#ff0000.**
- Por valor HSL (del inglés *Hue, Saturation, Lightness*; «matiz, saturación, luminosidad»). Indicado los valores correspondientes para cada componente (*Hue* entre 0° y 360°; *Saturation* entre 0% y 100%; *Lightness* entre 0% y 100%) en formato **hsl(*h, s, l*).** Por ejemplo, el color rojo se asigna con el valor **hsl(0, 100%, 50%).** Con la variante **hsla(*h, s, l, a*)** se añade un componente que indica la transparencia del color con un valor decimal comprendido entre 0 (completamente transparente) y 1 (completamente opaco).

A continuación, se muestran dos ejemplos de asignación de color sobre un mismo código HTML expresando dichos colores con diferentes notaciones.

Código HTML:

```
<div id="noticia">
El diario <cite>ABC</cite> redactó la increíble noticia de la siguiente
manera: "Un pie se descuelga por las escalerillas del módulo lunar posado en
el Mar de la Tranquilidad."
</div>
```


© Ediciones Paraninfo

Notación por valor RGB:

```
#noticia {
   padding: 1em;
   background-color: rgb(200,200,200);
}
```

El diario *ABC* redactó la increíble noticia de la siguiente manera: "Un pie se descuelga por las escalerillas del módulo lunar posado en el Mar de la Tranquilidad."

Figura 3.4. *Mediante RGB se pueden referenciar hasta un total de 16.777.216 colores distintos.*

Notación por nombre:

```
#noticia {
   padding: 1em;
   background-color: wheat;
}
```

El diario *ABC* redactó la increíble noticia de la siguiente manera: "Un pie se descuelga por las escalerillas del módulo lunar posado en el Mar de la Tranquilidad."

Figura 3.5. *La notación por nombre permite recordar los colores más fácilmente.*

3.5.5. Imágenes

Las referencias a imágenes admiten dos tipos de valores: archivos o degradados.

Los archivos se referencian utilizando la función **url()**. Esta referencia puede ser local o a un recurso externo.

Ejemplo:

Se asigna una imagen de fondo a un elemento ** **. La imagen de fondo del elemento ** ** con identificador **huertos** se obtiene de un fichero alojado en la misma carpeta en la que se encuentra el documento CSS.

```
<div id="huertos">
   <p>Los huertos ecológicos son cada vez más populares gracias a la calidad
de sus productos.</p>
</div>
```

© Ediciones Paraninfo

```
#huertos {
   padding: 1em;
   color:whitesmoke;
   font-weight: bolder;
   background-image: url("huerta.jpg");
}
```

Figura 3.6. *Las imágenes de fondo son un recurso gráfico muy llamativo.*

La misma imagen, podría estar alojada en un servidor web. En ese caso, la referencia en la hoja de estilos CSS tendrá el siguiente aspecto:

```
background-image: url("http://www.paraninfo.es/huerta.jpg");
```

Un degradado o gradiente de color técnicamente no tiene la consideración de color y no se puede aplicar como tal, sino que se considera una imagen, por lo que su utilización debe realizarse en el contexto adecuado. Para crear una imagen con un gradiente hay que utilizar la palabra **linear-gradient, radial-gradient, conic-gradient, repeating-linear-gradient** o **repeating-radial-gradient** e indicar los colores que utilizar, la dirección, el momento de la transición entre colores… Las posibilidades son casi infinitas y permiten a los diseñadores dar rienda suelta a su creatividad.

Ejemplo:

Se muestra el elemento ** ** con un gradiente de color de fondo.

```
<div id="noticia">
El diario <cite>La Vanguardia</cite> publicó la noticia en portada
concediéndole la importancia que se merecía: "Los habitantes de la Tierra, en
vela permanente, contemplan, a través de la Televisión, con asombro y emoción
como un hombre llamado Neil Armstrong baja por una escalera y pone el pie en
la Luna".
</div>
```

```
#noticia {
   padding: 1em;
   background: linear-gradient(to bottom, rgb(210,190,190), aquamarine);
}
```


© Ediciones Paraninfo

El diario *La Vanguardia* publicó la noticia en portada concediéndole la importancia que se merecía: "Los habitantes de la Tierra, en vela permanente, contemplan, a través de la Televisión, con asombro y emoción como un hombre llamado Neil Armstrong baja por una escalera y pone el pie en la Luna".

Figura 3.7. *Los gradientes son altamente configurables.*

3.5.6. Tipos de letras

En computación se conoce como *tipo de letra, tipografía* o *fuente* al conjunto de modelos gráficos que representan cada uno de los caracteres y símbolos representables por el ordenador y que se almacena en un fichero. Existe un número casi ilimitado de tipos de letras creados por los diseñadores gráficos y, como puede imaginarse, no están todos disponibles en todos los ordenadores.

Para poder disponer de un tipo de letra específico hay que tener el fichero que almacena la información gráfica en el ordenador o, en caso contrario, algún mecanismo para obtener dicha información en el momento de su uso.

Desde el punto de vista del diseño web, existen ciertas tipografías que se pueden utilizar de manera segura y general, ya que son comunes a todos los sistemas. Se denominan **tipos de letra seguros** y la lista no es muy extensa: **Arial, Courier New, Georgia, Times New Roman, Trebuchet MS** y **Verdana.** De manera específica CSS define cinco tipos de letra genéricos cuya representación puede no ser exactamente la misma en todos los sistemas. Estos tipos genéricos son: **serif, sans-serif, monospace, cursive** y **fantasy.**

Este texto está escrito con tipo de letra elegido por defecto por el navegador

Este texto está escrito con tipo de letra Courier New

Este texto está escrito con tipo de letra monospace

Figura 3.8. *Solo algunas tipografías están disponibles en todos los ordenadores.*

A partir de este conjunto de opciones básicas de tipos de letras, se abre un universo de múltiples y muy atractivas tipografías creadas por diseñadores y artistas. De forma general, los navegadores admiten formatos de letra diferentes, por lo que hay que especificar alternativas para todos ellos. El formato más común entendido por la mayoría de los navegadores es *The Web Open Font Format* en sus versiones 1 y 2 (extensiones woff y woff2). Además, un gran número de navegadores admiten los tipos de letra *TrueType Fonts* (extensión .ttf), *OpenType Fonts* (extensión .otf) y *SVG Fonts* (extensión .svg).

Se puede referenciar una fuente no genérica en la hoja de estilos CSS. Si está instalada en el ordenador, se mostrará correctamente. Si no está disponible, se mostrará con el tipo de letra por defecto.

© Ediciones Paraninfo

Este texto está escrito con un tipo de letra local

Este texto está escrito con un tipo de letra no disponible

Figura 3.9. *Los navegadores tienen un plan alternativo si el tipo de letra elegido no está disponible.*

Para evitar los inconvenientes surgidos por no tener instalado el tipo de letra en los ordenadores de los usuarios, CSS permite cargar la tipografía desde un servidor.

La carga de tipografías se hace con la regla @font-face. Tiene la siguiente sintaxis básica:

```
@font-face {
        font-family: alias-de-la-fuente;
        src: url("url-de-la-fuente");
}
```

El siguiente código CSS asigna al alias **fuente-local** una fuente almacenada en el fichero **Rubik-Bold.ttf.** Este fichero estará almacenado en el servidor y lo descargará el cliente cuando sea necesario, garantizando la disponibilidad del tipo de letra.

```
@font-face {
    font-family: fuente-local;
    src: url("fonts/Rubik-Bold.ttf");
}
```

Por supuesto, el fichero, con la tipografía, puede estar almacenado en un servidor distinto del que aloja a la página HTML, tal y como se puede ver en el siguiente código de ejemplo:

```
@font-face {
    font-family: 'Hanalei Fill';
    src: url("https://fonts.gstatic.com/s/hanaleifill/v9/
fC1mPYtObGbfyQznIaQzPQi8UAjA.woff2");
}
```

Otra alternativa para importar fuentes de letra externas a la página web consiste en utilizar la regla **@import.** Esta regla permite obtener un recurso a partir de una URL.

En el siguiente código se muestra la regla de carga y el uso de una fuente de letra:

```
@import url('https://fonts.googleapis.com/css2?family=Roboto');

body {
    font-family: 'Roboto', sans-serif;
}
```


© Ediciones Paraninfo

Debido a que el formato del fichero que contiene el tipo de letra puede no ser compatible con el navegador, es recomendable proporcionar alternativas que incluyan varios formatos distintos.

Nota técnica

No todas las fuentes son gratuitas. Asegúrate de que tienes los derechos adecuados sobre las fuentes que uses en tus páginas y aplicaciones web.

3.6. Selectores CSS

Los selectores constituyen el mecanismo que determina sobre qué elemento o elementos se debe aplicar un estilo. Es un sistema muy elaborado y capaz de conseguir un alto nivel de precisión a la hora de determinar sobre qué propiedades de qué elementos se van a aplicar los valores deseados.

Se puede considerar los selectores como una especie de cribas de diferentes calibres, capaces de permitir o impedir el paso de los elementos en función de su tamaño. En el caso concreto de CSS, el tamaño del elemento no es relevante, pero los elementos de HTML disponen de otros atributos que pueden participar de este proceso de selección; solo a aquellos elementos que «pasen el filtro» se les aplicará el estilo correspondiente.

Figura 3.10. *CSS aplica el estilo a un elemento si este encaja con el selector.*

Los selectores tienen un nivel de especificidad dispar, desde muy generales hasta muy específicos. Lo mismo ocurre con el nivel de complejidad; hay selectores muy sencillos y fáciles de comprender y otros más elaborados y complejos.

Se pueden agrupar los selectores como básicos, combinadores, pseudoclases y pseudoelementos.

© Ediciones Paraninfo

3.6.1. Selectores básicos

Son los selectores más sencillos de entender y de uso más habitual. Se detallan en la Tabla 3.5.

Tabla 3.5. Selectores básicos

Nombre	Selecciona	Sintaxis
Universal	Todos los elementos.	*
De tipo	Todos los elementos de un tipo determinado.	*nombreElemento*
De clase	Todos los elementos de una determinada clase.	*.nombreClase*
De identificador	El elemento con el identificador indicado.	*#identificador*
De atributo	Los elementos que tienen un determinado atributo con unas características específicas. = → El atributo tiene un determinado valor. *= → El atributo contiene un determinado valor. ^= → El atributo comienza con un determinado valor. $= → El atributo termina con un determinado valor. ~= → El atributo contiene un determinado valor en una lista de valores.	[*atributo*] *elemento*[*atributo*] *elemento*[*atributo*="*cadena*"] *elemento*[*atributo**="*cadena*"] *elemento*[*atributo*^="*cadena*"]] *elemento*[*atributo*$="*cadena*"]] *elemento*[*atributo*~="*cadena*"]]

Ejemplo:

Aquí se utilizan selectores de tipo universal, de elemento, de clase, de identificador y de atributo.

```
<body>
<h1>RESUMEN ANUAL</h1>
<form action="">
   <input type="text" placeholder="Departamento">
   <input type="number" placeholder="Límite de ventas">
</form>
<p id="introduccion">
   Durante la temporada los resultados han sido satisfactorios...
</p>
<table>
   <tr>
      <th>Ingresos</th>
      <th>Gastos</th>
      <th>Diferencia</th>
   </tr>
  <tr>
     <td>100</td>
```


© Ediciones Paraninfo

```
      <td>50</td>
      <td class="positivo">50</td>
   </tr>
   <tr>
      <td>200</td>
      <td>190</td>
      <td class="position">10</td>
   </tr>
   <tr>
      <td>80</td>
      <td>180</td>
      <td class="negativo">-100</td>
   </tr>
</table>
</body>
```

```
/* Selector universal */
* {
   margin:3px;
}

/* Selector de elemento */
h1 {
   color: rgb(50, 50, 50);
   font-style: italic;
}

/* Selector de identificador */
#introduccion {
   border-style: dotted;
   padding: 1em;
}

/* Selectores de clase */
.positivo {
   color:green;
   font-weight: bolder;
}
.negativo {
   color: red;
   font-weight: bolder;
}

/* Selectores de atributo */
input[type]{
   background-color: rgb(196, 248, 196);
}
input[type=text]{
   color: rgb(0, 98, 255);
   font-weight: bolder;
}
```

© Ediciones Paraninfo

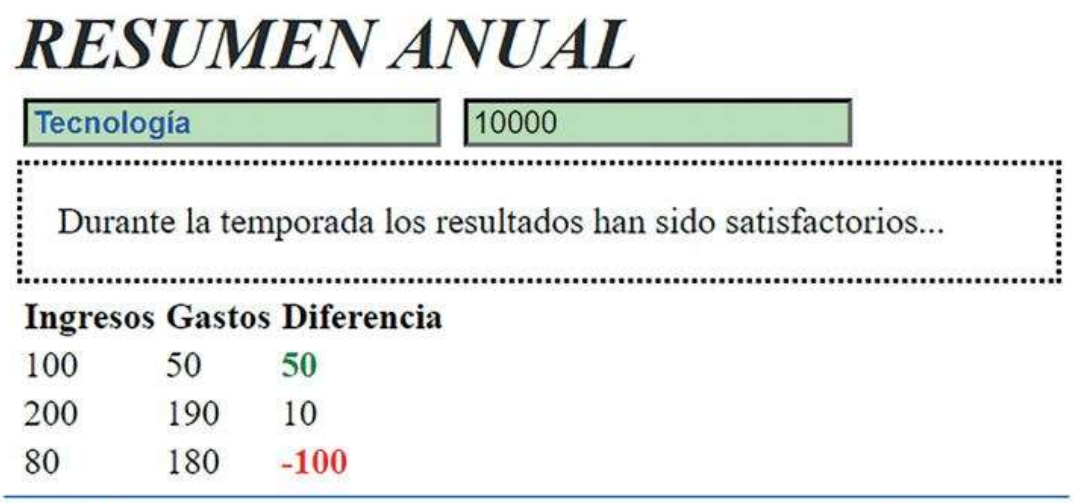


Figura 3.11. *Conocer los selectores es fundamental para poder aplicar los estilos CSS.*

3.6.2. Agrupación de selectores

En ocasiones se desea aplicar el mismo estilo a varios elementos. Una posible solución consiste en crear un selector distinto para cada uno de ellos con las mismas propiedades y los mismos valores, aunque esta solución puede no ser la mejor.

CSS permite crear reglas con selectores múltiples pudiendo así aplicar las mismas declaraciones a distintos grupos de elementos.

La sintaxis es:

```
elemento, elemento, elemento { propiedad : valor }
```

Ejemplo:

Creación de un selector que aplique el mismo estilo a varios elementos.

```
<div>
<h1>CSS</h1>
<p>Lorem ipsum dolor sit amet consectetur adipisicing elit.</p>
<h2>Introducción</h2>
<p>Consequuntur veniam, quaerat repellendus facilis laborum magni
reprehenderit.</p>
<h2>Breve historia</h2>
<p>Doloribus fugit, nesciunt sequi pariatur adipisci rem nostrum optio...</p>
</div>
```

```
h1, h2, h3 {
color: blue;
}
```


© Ediciones Paraninfo

CSS

Lorem ipsum dolor sit amet consectetur adipisicing elit.

Introducción

Consequuntur veniam, quaerat repellendus facilis laborum magni reprehenderit.

Breve historia

Doloribus fugit, nesciunt sequi pariatur adipisci rem nostrum optio...

Figura 3.12. *Al agrupar selectores se reduce el tamaño de las hojas de estilo.*

3.6.3. Combinadores

Son selectores en los que se tiene en cuenta la relación entre elementos en la estructura jerárquica del documento.

Tabla 3.6. Combinadores

Nombre	Combinador	Descripción
De hermanos	A ~ B	A y B son hermanos.
De hijos	A > B	B es hijo de A.
De hermanos adyacentes	A + B	A y B son hermanos y B está inmediatamente a continuación de A.
De descendientes	A B	B es descendiente de A, pero no necesariamente es hijo directo.

Ejemplo:

Dado el siguiente código HTML:

```
<div>
   <div>
      <h2>Sistema operativo</h2>
      <p>El sistema operativo es el componente...</p>
   </div>
   <article>
      <p>Un buen antivirus protege al ordenador...</p>
   </article>
</div>
```

Esta regla centra el texto de los elementos **<p>** que sean hijos de un elemento **<article>**.

```
article > p {
   text-align: center;
}
```

© Ediciones Paraninfo

Sistema operativo

El sistema operativo es el componente...

Un buen antivirus protege al ordenador...

Figura 3.13. *Los combinadores son más complejos de usar, pero proporcionan mayor nivel de precisión.*

Esta regla centrará el texto de los elementos **<p>** que sean hermanos de un elemento **<h2>**.

```
h2 ~ p {
   text-align: center;
}
```

Sistema operativo

El sistema operativo es el componente...

Un buen antivirus protege al ordenador...

Figura 3.14. *Los selectores creados con combinadores posibilitan múltiples alternativas.*

Esta regla centrará el texto de los elementos **<p>** que sean descendientes de un elemento ** **.

```
div p {
   text-align: center;
}
```

Sistema operativo

El sistema operativo es el componente...

Un buen antivirus protege al ordenador...

Figura 3.15. *Prácticamente todas las relaciones entre elementos se pueden expresar con combinadores.*

3.6.4. Pseudoclases

Se añade como palabra clave a continuación un selector convencional, creando un nuevo filtro de selección en función del estado del elemento o de los elementos filtrados por el selector.

La sintaxis es:

```
selector:pseudoclase { propiedad: valor; }
```

En la Tabla 3.7 se muestra una extensa relación de las pseudoclases existentes. Se omiten aquellas que son experimentales y no tienen un soporte generalizado por parte de los navegadores o cuyo uso es poco habitual.

© Ediciones Paraninfo

Tabla 3.7. Pseudoclases

Nombre	Estado del elemento o elementos proporcionados por el selector
:active	El elemento ha sido activado por el usuario.
:checked	Afecta a elementos **<input>** de tipo **radio** o **checkbox** o a elementos **<option>** cuando han sido marcados.
:default	Elemento de un formulario marcado como predeterminado.
:disabled	El elemento está deshabilitado.
:empty	El elemento no tiene hijos.
:enabled	El elemento está habilitado.
:first-child	El primer elemento de entre un grupo de elementos hermanos.
:first-of-type	El primer elemento de un tipo de entre un grupo de elementos hermanos.
:focus	El elemento (de un formulario) tiene el foco.
:focus-within	El elemento (de un formulario) o uno de los elementos contenidos tiene el foco.
:hover	El cursor del ratón se encuentra sobre el elemento.
:indeterminate	El estado es indeterminado. Aplica a los elementos **<input>** de tipo **checkbox** y **radio** y a los elementos **<progress>.**
:in-range	El valor de un campo **<input>** se encuentra dentro del rango indicado por los atributos **min** y **max.**
:invalid	El elemento (de un formulario) es válido.
:lang(*idldioma*)	El elemento está en un idioma determinado. Por ejemplo **:lang(es)**
:last-child	El último elemento de un grupo de elementos hermanos.
:last-of-type	El último elemento de un tipo de entre un grupo de elementos hermanos.
:link	El enlace referenciado por el elemento no ha sido visitado.
:not()	Los elementos que no coinciden con una lista de selectores.
:nth-child(*n*)	El elemento en la posición *n* de un grupo de elementos hermanos. Se puede utilizar **odd** para indicar las posiciones pares y **even** para las posiciones pares en el parámetro **n.**
:nth-last-child(*n*)	El elemento en la posición *n* de un grupo de elementos hermanos, contando desde el final.
:nth-last-of-type(*n*)	El elemento de un tipo en la posición *n* de un grupo de elementos hermanos contando desde el final.
:nth-of-type(*n*)	El elemento de un tipo en la posición *n* de un grupo de elementos hermanos.
:only-child	El elemento no tiene elementos hermanos.
:only-of-type	El elemento no tiene elementos hermanos de un determinado tipo.

© Ediciones Paraninfo

Nombre	Estado del elemento o elementos proporcionados por el selector
:optional	El elemento (de tipo **<input>, <select>** o **<textarea>**) no tiene el atributo **required.**
:out-of-range	El valor de un campo **<input>** se encuentra fuera del rango indicado por los atributos **min** y **max.**
:read-only	El elemento no es editable.
:read-write	El elemento es editable.
:required	El elemento es obligatorio.
:root	El elemento de más alto nivel del árbol que representa la estructura de la página.
:target	La zona (por ejemplo, un ** **), que es referenciada desde un enlace interno.
:valid	El elemento (de un formulario) no es válido.
:visited	El enlace referenciado por el elemento ha sido visitado.

Ejemplo:

Utilizando pseudoclases, se colorea el primer elemento de la lista en rojo, el último en azul y el elemento que está en la posición 3 en verde.

```
<ol>
   <li>Plataformas</li>
   <li>Simulador</li>
   <li>Deportes</li>
   <li>Estrategia</li>
   <li>Aventura gráfica</li>
</ol>
```

```
li {
   font-weight: bolder;
}

li:first-child {
   color:red;
}

li:nth-child(3) {
   color:green;
}

li:last-child {
   color: blue;
}
```


© Ediciones Paraninfo

1. Plataformas
2. Simulador
3. Deportes
4. Estrategia
5. Aventura gráfica

Figura 3.16. *Las pseudoclases proporcionan reglas relacionadas con la posición.*

Actividad resuelta 3.1

Asignación de estilo CSS

Asignar estilo a los elementos de la tabla que se define en el código HTML proporcionado, prestando especial interés a los selectores utilizados. Los requisitos son los siguientes:

- Al encabezado y al pie se les asigna un color marrón de fondo, un color de texto blanco, y se convierte el texto a mayúsculas.
- A la última columna se le pone el texto en negrita.
- A las filas impares del cuerpo de la tabla se les asigna un color de fondo añil y un color de texto blanco.
- A las filas pares del cuerpo de la tabla se les asigna un color de fondo blanco y un color de texto añil.

```
<table>
   <thead>
      <tr>
      <td>Concepto</td>
      <td>Primer semestre</td>
      <td>Segundo semestre</td>
      <td>Total</td>
      </tr>
   </thead>
   <tbody>
      <tr>
         <td>Alimentación</td>
         <td>1500</td><td>1750</td><td>3250</td>
      </tr>
      <tr>
         <td>Tecnología</td>
         <td>1500</td><td>1750</td><td>3250</td>
      </tr>
      <tr>
         <td>Deporte</td>
         <td>1500</td><td>1750</td><td>3250</td>
      </tr>
      <tr>
        <td>Calzado</td>
         <td>1500</td><td>1750</td><td>3250</td>
      </tr>
      <tr>
        <td>Viajes</td>
```

© Ediciones Paraninfo

```
        <td>1500</td><td>1750</td><td>3250</td>
      </tr>
      <tr>
         <td>Papelería</td>
         <td>1500</td><td>1750</td><td>3250</td>
      </tr>
   </tbody>
   <tfoot>
      <tr>
         <td>Total</td>
         <td>0000</td><td>0000</td><td>0000</td>
      </tr>
   </tfoot>
</table>
```

Solución

```
thead>tr, tfoot>tr{
   background-color: brown;
   color: white;
   text-transform: uppercase;
}

td:last-child{
   font-weight: bolder;
}

tbody>tr:nth-child(odd) {
   background-color: rgb(28, 76, 150);
   color: white;
}

tbody>tr:nth-child(even) {
   background-color:white;
   color: rgb(28, 76, 150);
}
```

CONCEPTO	PRIMER SEMESTRE	SEGUNDO SEMESTRE	TOTAL
Alimentación	1500	1750	3250
Tecnología	1500	1750	3250
Deporte	1500	1750	3250
Calzado	1500	1750	3250
Viajes	1500	1750	3250
Papelería	1500	1750	3250
TOTAL	0000	0000	0000

Figura 3.17. *Con las pseudoclases se puede diseñar espectaculares tablas de datos.*

© Ediciones Paraninfo

3.6.5. Pseudoelementos

Permiten seleccionar elementos por reglas que no forman parte de la estructura general del documento HTML. Al igual que las pseudoclases, se añaden a los selectores, pero no para crear un nuevo nivel de filtrado, sino para seleccionar partes más precisas del conjunto de elementos proporcionados por el selector. La separación entre el selector y el pseudoelemento se suele hacer con dos puntos dobles (::) para distinguirlos de las pseudoclases, pero los navegadores entienden también los dos puntos simples (:).

La sintaxis es:

```
selector::pseudo-elemento { propiedad: valor; }
```

En la Tabla 3.8 se muestra una tabla con los pseudoelementos de uso más extendido.

Tabla 3.8. Pseudoelementos

Nombre	Descripción
::after	Permite añadir contenido con el atributo **content** después del elemento seleccionado.
::before	Permite añadir contenido con el atributo **content** antes del elemento seleccionado.
::first-letter	Primera letra del primer bloque de un elemento seleccionado.
::first-line	Primera línea del primer bloque de un elemento seleccionado.
::selection	Permite modificar el estilo de la selección realizada con el ratón.

Ejemplo:

Utilizando pseudoelementos:

- Añadir el texto «SEGUIR LEYENDO…» inmediatamente después del elemento con **id** «**ElQuijote**».
- Asignar **3em** como tamaño de la primera letra del elemento con **id** «**ElQuijote**».

```
<p id="ElQuijote">
En un lugar de La Mancha, de cuyo nombre no quiero acordarme,
no ha mucho tiempo que vivía un hidalgo
</p>
```

```
#ElQuijote:after {
   content: "SEGUIR LEYENDO...";
}
#ElQuijote::first-letter {
  font-size: 3em;
}
```

© Ediciones Paraninfo

En un lugar de la Mancha, de cuyo nombre no quiero acordarme, no ha mucho tiempo que vivía un hidalgo... Es, pues, de saber que este sobredicho hidalgo, los ratos que estaba ocioso —que eran los más del año—, SEGUIR LEYENDO...

Figura 3.18. *Con los pseudoelementos el nivel de granularidad del selector es muy alto.*

Actividad propuesta 3.3

Selectores

Dado el bloque de código HTML siguiente, escribe los selectores que se indican a continuación:

- Selector de todos los elementos <h2>.
- Selector del primer elemento <h2>.
- Selector del último elemento <h2>.
- Selector del elemento con identificador #principal.
- Selector de los elementos <h2> que son descendientes del elemento con identificador #principal.
- Selector de los elementos de la clase .Articulo.
- Selector de los elementos <h2> que son descendientes de los elementos de la clase .Articulo.
- Selector del último hijo de los elementos <article> que son descendientes del elemento con identificador #secundario.

```
<h2>Título de la sección</h2>
<div id="principal">
    Texto div principal
    <article class="Articulo">
        <h2>Artículo 1</h2>
        <p>Texto del artículo 1</p>
    </article>
    <article class="Articulo">
        <h2>Artículo 2</h2>
        <p>Texto del artículo 2</p>
    </article>
</div>
<div id="secundario">
    Texto div secundario
    <article class="Articulo">
        <h2>Artículo 3</h2>
        <p>Texto del artículo 3</p>
    </article>
    <article class="Articulo">
        <h2>Artículo 4</h2>
        <p>Texto del artículo 4</p>
    </article>
</div>
```


© Ediciones Paraninfo

3.7. Propiedades CSS

En CSS es posible modificar prácticamente cualquier aspecto del estilo de un elemento de HTML, ya que el conjunto de propiedades es muy extenso y completo.

En esta unidad se tratan aspectos relativos al funcionamiento del modelo de maquetación utilizado en CSS, conocido como «modelo de cajas». Además, se muestran algunas de las propiedades y técnicas más habituales.

Es conveniente recordar que, además de las propiedades específicas de cada elemento, existen propiedades comunes que tienen relación con la herencia y la aplicación en cascada de las reglas CSS.

Recuerda

Las propiedades **inherit, initial, unset** y **revert** son comunes y afectan a todos los elementos, permitiendo definir su relación con los elementos que los contienen.

3.7.1. Contenedores y posicionamiento: el modelo de cajas

La maquetación en CSS se basa en el denominado «modelo de cajas». Según este concepto, todos los elementos se encuentran ubicados en un contenedor rectangular denominado **caja.**

Estas cajas tienen un conjunto de propiedades básicas comunes que afectan directamente a su ubicación en la página y al espacio que ocupan:

- **Margen exterior.** Es el espacio vacío que se agrega al elemento y que lo separa del resto de elementos circundantes. Su dimensión se modifica con la propiedad **margin** y sus variantes **margin-top, margin-botton, margin-left** y **margin-right.**
- **Margen interior.** Es el espacio vacío que se agrega al elemento y separa del borde a su contenido. Su dimensión se modifica con la propiedad **padding** y sus variantes **padding-top, padding-bottom, padding-left** y **padding-right.**
- **Borde.** Es un espacio que puede tener distintos tipos de relleno y que se encuentra entre el margen exterior y el margen interior. Se configura con la propiedad **border** y ocupa espacio, por lo que aumenta el tamaño del elemento. Algunas propiedades para configurar los bordes son **border-radius, border-style, border-width** y **border-color.**
- **Contorno.** Es similar al borde, pero no ocupa espacio ya que se dibuja encima del elemento. Se configura con las propiedades **outline, outline-color, outline-width** y **outline-style.**
- **Ancho.** Representa el ancho del espacio del contenedor que puede contener elementos. Se modifica con la propiedad **width.**
- **Alto.** Representa el alto del espacio del contenedor que puede contener elementos. Se modifica con la propiedad **height.**

© Ediciones Paraninfo

En la imagen de la Figura 3.19, proporcionada por el inspector del Google Chrome en las «Herramientas para desarrolladores», se puede observar un ejemplo de qué espacio ocupan en un elemento cada una de las partes que componen una caja.

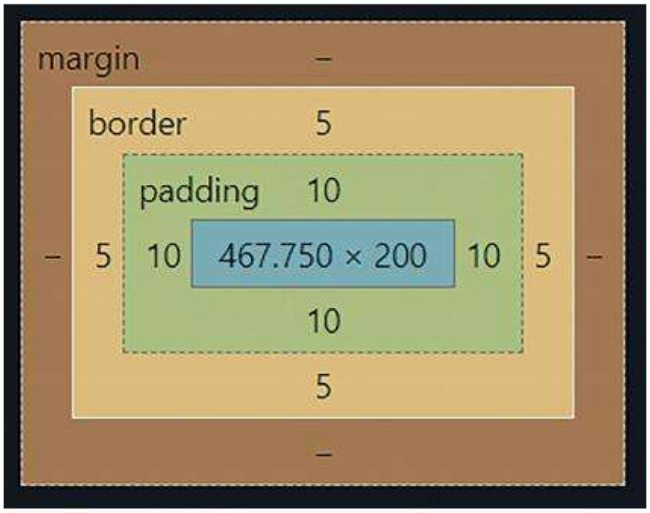


Figura 3.19. *Las propiedades margin, border y padding afectan a distintos aspectos del espacio del elemento.*

La Figura 3.19 se basa en los datos del siguiente ejemplo: se aplica a un elemento ** ** vacío un estilo en el que se define un ancho porcentual del 50% del espacio disponible, un alto absoluto de 200 píxeles, un margen exterior absoluto de 0 píxeles, un margen interior absoluto de 10 píxeles y un borde de 5 píxeles de ancho.

```
<div id="divExterior">
    <div id="divInterior"></div>
</div>
```

```
#divInterior{
    background-color: rgb(255, 148, 25);
    width: 100%;
    height: 100%;
}

#divExterior {
    width: 50%;
    height: 200px;
    margin: 0px;
    padding:10px;
    border-width: 5px;

    border-style: solid;
    border-color:brown;
    background-color:rgb(250, 230, 120);
}
```


© Ediciones Paraninfo

El resultado se puede observar en la Figura 3.20. Pese a que el margen indicado en el estilo no tiene grosor, se aprecia un margen entre el ** ** y el navegador: el margen del propio elemento **<body>**.

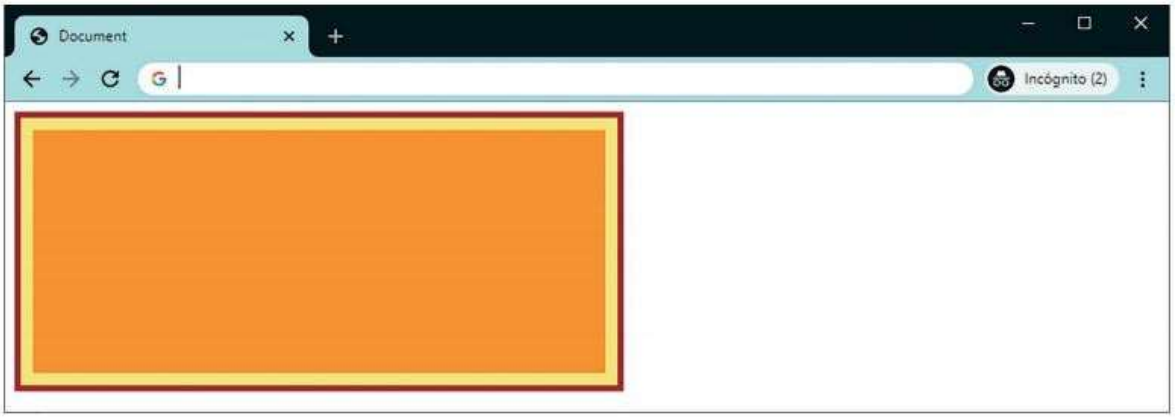


Figura 3.20. *El modelo de cajas proporciona control total sobre el espacio.*

Como se ha comentado anteriormente, los márgenes izquierdos y derechos se pueden asignar de manera independiente a cada uno de los lados:

```
#divExterior {
    width: 50%;
    height: 200px;
    margin: 0px;
    margin-left: 30px;
    padding:10px;
    padding-bottom: 50px;
    border-width: 5px;

    border-style: solid;
    border-color:brown;
    background-color:rgb(250, 230, 120);
}
```

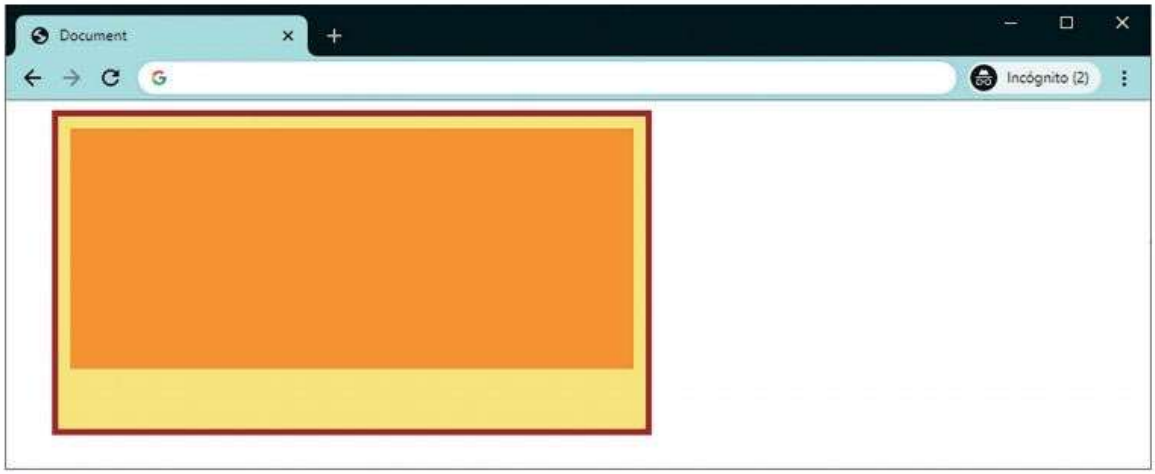


Figura 3.21. *Border, padding y margin tienen variantes para determinar sobre qué lado de la caja actuar.*

© Ediciones Paraninfo

Nota técnica

Para eliminar los márgenes del navegador y hacer que los contenidos aparezcan pegados a los bordes de este, se puede usar la siguiente regla:

```
body {
    margin:0px;
}
```

Hay dos tipos de cajas: **cajas en bloque** y **cajas en línea** (también llamadas de tipo *inline*). El que una caja sea de un tipo u otro tiene implicaciones directas en cómo se comporta la caja en relación con la página y con las demás cajas.

Las características fundamentales de las cajas en bloque son las siguientes:

- Después de la caja se realiza un salto de línea.
- Suele ocupar el 100 % del espacio del contenedor en el que se encuentra.
- Se aplican los valores de las propiedades **width** y **height.**
- Los márgenes, bordes y rellenos desplazan al resto de cajas.

Por su parte, las características principales de las cajas en línea son estas:

- No generan salto de línea. Por defecto, ocupan el espacio que ocupa su contenido.
- No se aplican los valores de las propiedades **width** y **height.**
- Los márgenes, bordes y rellenos horizontales desplazan al resto de cajas en línea.
- Los márgenes, bordes y rellenos verticales no desplazan al resto de cajas en línea.

El siguiente código HTML contiene ejemplos de cajas en línea (elemento ****) y en bloque (elemento ** **).

```
<div>Caja de tipo bloque</div>
<div>Caja de tipo <span>en línea</span> dentro de una caja de tipo bloque</
div>
```

Como se puede observar en la Figura 3.22, el primer elemento ** ** provoca un salto de línea mientras que el elemento **** no lo hace.

Caja de tipo bloque
Caja de tipo en línea dentro de una caja de tipo bloque

Figura 3.22. *Los elementos de bloque generan un salto de línea, los elementos de línea no.*

Todos los elementos de HTML pertenecen a uno de estos dos tipos por defecto, pero esta propiedad se puede modificar utilizando la propiedad **display,** asignándole los valores **inline** o **block** según se desee.

Por ejemplo, asignando al código HTML anterior una modificación de la propiedad **display,** el elemento **** comenzará a comportarse como un elemento de tipo caja.

© Ediciones Paraninfo

```
span {
   display: block;
}
```

Caja de tipo bloque
Caja de tipo
en línea
dentro de una caja de tipo bloque

Figura 3.23. *Con la propiedad display, se puede modificar el comportamiento por defecto de los elementos.*

La relación de valores de la propiedad **display** es muy extensa, permitiendo incluso combinarlos. Se dividen en aquellos orientados a la visualización **interna** (cómo se distribuyen los componentes **dentro** de la caja) y los orientados a la visualización **externa** (cómo se comporta la caja en relación con los elementos que la rodean). Además, existe el valor especial **none** que desactiva todas las modificaciones de la propiedad **display** que afecten al elemento en cuestión y a todos los descendientes.

Los valores más utilizados para el tipo de visualización interna son los siguientes:

- **flex.** El elemento se comporta como un elemento de bloque y de acuerdo con el modelo denominado ***flexbox***. Mediante la propiedad **flex-direction** se indica en qué dirección se desea que «fluya» el contenido del contenedor. A un contenedor con la propiedad **display** con valor **flex** se le denomina **contenedor flex.**
- **grid.** El elemento se comporta como un elemento de bloque y de acuerdo con el modelo de cuadrícula.

Ejemplos de aplicación del tipo de **display** interno **flex** con diferentes valores para la propiedad **flex-direction:**

```
<div id="divFlex">
    <div>Primer contenedor</div>
    <div>Segundo contenedor</div>
    <div>Tercer contenedor</div>
</div>
```

```
#divFlex {
    display: flex;
    flex-direction: row;
}
```

Primer contenedor	Segundo contenedor	Tercer contenedor

Figura 3.24. *El modelo flexbox facilita la alineación de los elementos.*

© Ediciones Paraninfo

```
#divFlex {
    display: flex;
    flex-direction: row-reverse;
}
```

Tercer contenedor	Segundo contenedor	Primer contenedor

Figura 3.25. *Con flexbox, se puede invertir el orden natural de los elementos.*

```
#divFlex {
    display: flex;
    flex-direction: column;
}
```

Primer contenedor
Segundo contenedor
Tercer contenedor

Figura 3.26. *Flexbox trabaja en los ejes horizontal y vertical.*

```
#divFlex {
    display: flex;
    flex-direction: column-reverse;
}
```

Tercer contenedor
Segundo contenedor
Primer contenedor

Figura 3.27. *Flexbox proporciona la misma flexibilidad en vertical que en horizontal.*

Nota

El módulo de caja flexible o *flexbox* es un modelo de diseño que proporciona una gran versatilidad. En el siguiente enlace se puede obtener información detallada sobre este módulo de CSS3:

https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_Flexible_Box_Layout

© Ediciones Paraninfo

Ejemplos de aplicación del tipo de **display** interno **grid** con diferentes valores para la propiedad **grid-template-colums:**

```
<div id="divGrid">
    <div>Primer contenedor</div>
    <div>Segundo contenedor</div>
    <div>Tercer contenedor</div>
    <div>Cuarto contenedor</div>
    <div>Quinto contenedor</div>
    <div>Sexto contenedor</div>
</div>
```

```
#divGrid {
    display:grid;
    grid-template-columns: auto auto auto;
}
```

Primer contenedor	Segundo contenedor	Tercer contenedor
Cuarto contenedor	Quinto contenedor	Sexto contenedor

Figura 3.28. *El tipo de display grid distribuye los elementos como una tabla.*

```
#divGrid {
    display:grid;
    grid-template-columns: 100px 100px 100px;
}
```

Primer contenedor	Segundo contenedor	Tercer contenedor
Cuarto contenedor	Quinto contenedor	Sexto contenedor

Figura 3.29. *El tipo grid permite controlar el ancho de los elementos.*

```
#divGrid {
    display:grid;
    grid-template-columns: 70% 30%;
}
```

Primer contenedor	Segundo contenedor
Tercer contenedor	Cuarto contenedor
Quinto contenedor	Sexto contenedor

Figura 3.30. *La distribución de elementos por filas y columnas facilita la maquetación.*

© Ediciones Paraninfo

Algunos de valores más utilizados para el tipo de visualización externa son los siguientes:

- **block.** El elemento se comporta como un elemento de bloque.
- **inline.** El elemento se comporta como un elemento en línea.

Referencia

Para profundizar en el conocimiento del parámetro **style** se pueden utilizar los siguientes enlaces:

W3C:

https://www.w3.org/TR/css-display-3/

MDN (Mozilla Development Network):

https://developer.mozilla.org/es/docs/Web/CSS/display

3.7.2. Propiedad float

Por defecto, los elementos de bloque tienen un comportamiento que genera grandes limitaciones a la hora de maquetar las páginas web. Por suerte, este comportamiento se puede modificar mediante los estilos CSS.

Los elementos de bloque que habitualmente se utilizan para maquetar son <header>, <footer>, <aside>, <nav>, <article>, <section> y , siendo este último el más genérico, ya que no tiene información semántica asociada. Es importante conocer el comportamiento por defecto de estos elementos y saber realizar las modificaciones adecuadas para que se distribuyan de la forma correcta.

Utilizando el elemento como referencia, en esta sección se muestran algunas técnicas relacionadas con la maquetación en las que se utiliza la propiedad **float** de los elementos de bloque.

Se puede ilustrar el comportamiento por defecto de los elementos de bloque con un sencillo ejemplo. Las siguientes reglas asignan colores distintos a tres identificadores:

```
.contenedor {
    color:white;
}
#contenedor1 {
    background-color: red;
}
#contenedor2 {
    background-color: green;
}
#contenedor3 {
    background-color: blue;
}
```


© Ediciones Paraninfo

Los elementos de bloque, por defecto, ocupan todo el ancho del contenedor en el que se encuentran y se colocan unos debajo de otros, ya que insertan un salto de línea.

```
<div id="contenedor1" class="contenedor">Contenido del primer contenedor</div>
<div id="contenedor2" class="contenedor">Contenido del segundo contenedor</div>
<div id="contenedor3" class="contenedor">Contenido del tercer contenedor</div>
```

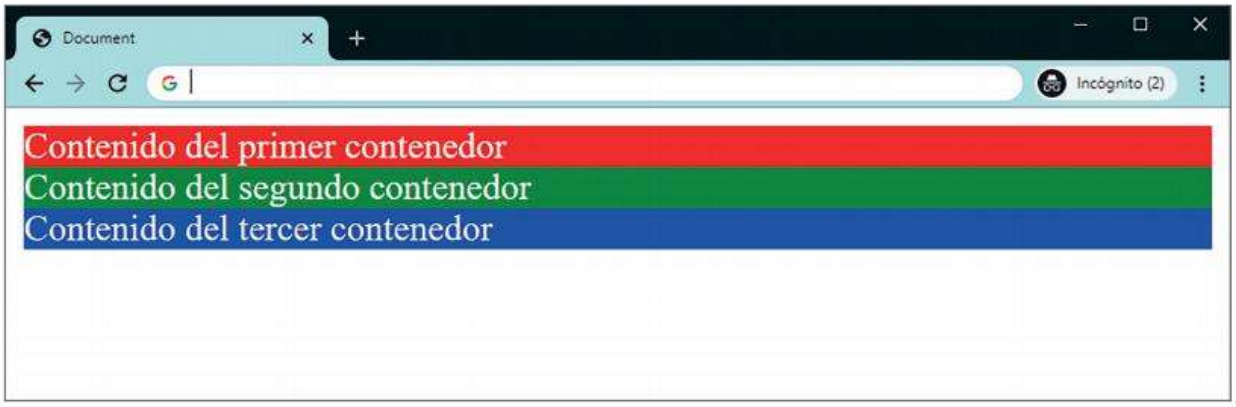


Figura 3.31. *Por defecto, los elementos de bloque ocupan todo el ancho del espacio disponible.*

Si se les asigna un ancho, ya sea en valores absolutos o relativos, los elementos de bloque siguen colocándose de forma vertical.

```
.contenedor {
    color:white;
    width: 30%;
}
```

Figura 3.32. *Por defecto, los elementos de bloque se posicionan uno debajo de otro, según el orden de aparición en el documento HTML.*

Este comportamiento por defecto se puede modificar mediante la propiedad **float.** Esta propiedad posiciona el elemento al lado izquierdo o derecho del contenedor en el que se encuentra. Los principales valores que admite son:

© Ediciones Paraninfo

- **left:** «empuja» el elemento hacia el lado izquierdo de su contenedor.
- **right:** «empuja» el elemento hacia el lado derecho de su contenedor.
- **none:** no se «empuja» el elemento. Es el valor por defecto.
- **inherit:** hereda el valor de la propiedad **float** del contenedor padre.

Por ejemplo, modificando el estilo se puede forzar a que el elemento **contenedor1** se desplace hacia la izquierda hasta el límite de su contenedor, el elemento **contenedor2** se desplace hacia la derecha hasta el límite de su contenedor y el elemento **contenedor3** se desplace hacia la derecha hasta encontrarse con el elemento **contenedor2.**

```
#contenedor1 {
    background-color: red;
    float:left;
}
#contenedor2 {
    background-color: green;
    float:right;
}
#contenedor3 {
    background-color: blue;
    float:right;
}
```

Figura 3.33. *Con la propiedad float se puede cambiar el comportamiento de los elementos de bloque.*

Si se modifica la regla aplicada a **contenedor3** dando a la propiedad **float** el valor **none**, dicho contenedor no flotará y se posicionará en su ubicación natural:

```
#contenedor3 {
    background-color: blue;
    float:none;
}
```

Nota técnica

Se puede centrar un elemento de tipo bloque fijándole un ancho y estableciendo **auto** como valor del atributo **margin.**

© Ediciones Paraninfo

Figura 3.34. *El valor none de la propiedad float indica que el elemento no debe flotar.*

3.7.3. Propiedad position

La propiedad **position** determina cómo se posiciona un elemento en el documento. Es una propiedad muy útil para la maquetación, ya que proporciona una gran flexibilidad.

Los valores que admite esta propiedad se muestran en la Tabla 3.9.

Tabla 3.9. Valores admitidos por la propiedad position

Valor	Descripción
static	Es el valor por defecto. El elemento se posiciona siguiendo el flujo normal del documento. No le afectan las propiedades **top, bottom, left, right** y **z-index.**
relative	El elemento se posiciona siguiendo el flujo normal del documento, pero se puede desplazar de forma relativa a su posición por defecto utilizando las propiedades **top, bottom, left** y **right.** El desplazamiento no desplaza a los demás elementos, por lo que puede proporcionar apilamiento que deberá organizarse utilizando la propiedad **z-index.**
absolute	El elemento deja de seguir el flujo normal del documento. Mediante las propiedades **top, bottom, left** y **right** el elemento se posiciona de manera relativa a su ancestro. Si no tiene ancestro, se posiciona de manera relativa a la página completa.
fixed	El elemento deja de seguir el flujo normal del documento. Mediante las propiedades **top, bottom, left** y **right** el elemento se posiciona de manera relativa a la página completa.
sticky	El elemento se posiciona siguiendo el flujo normal del documento. Mediante las propiedades **top, bottom, left** y **right** se fijan unas posiciones límites relativas a su contenedor, de tal manera que no excederá de estas.

Ejemplo:

A continuación se hace uso del posicionamiento relativo para centrar un dentro de otro.

El principal (el que tiene el atributo **id** con valor «div1») ocupa todo el ancho disponible y tiene fijada una altura fija en píxeles. El tamaño y la posición del interior (el que tiene el atributo **id** con valor «div2») se declara relativo al elemento dentro del que se encuentra: el ancho es el 50 % del espacio de su contenedor; la posición es de 10

© Ediciones Paraninfo

píxeles desde la parte superior del contenedor y de un 25 % del ancho del contenedor desde el lado izquierdo.

```
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <link rel="stylesheet" href="position-relative.css">
    <title>Document</title>
</head>
<body>
    <div id="div1">
        <div id="div2">Texto del div con posición relativa</div>
    </div>
</body>
</html>
```

```
#div1 {
    width: 100%;
    height: 200px;
    background-color: aquamarine;

    position: static;
}
#div2 {
    width: 50%;
    height: 180px;
    background-color:lightgray;

    position: relative;
    top:10px;
    left:25%;
}
```

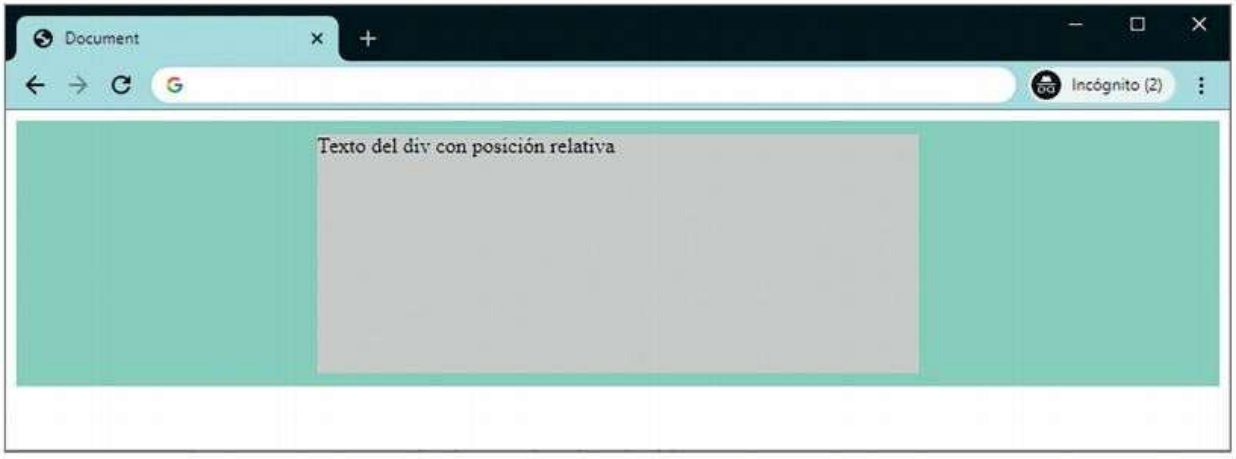


Figura 3.35. *Con el posicionamiento relativo los elementos se posicionan en función de su contenedor.*

© Ediciones Paraninfo

Actividad resuelta 3.2

Creación de un pie de página fijo

Crear las reglas CSS para realizar, dado el código HTML que se muestra a continuación, un pie de página que permanezca fijo en la parte inferior del navegador, aunque se realice un desplazamiento *(scroll)* vertical sobre la página.

```
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <link rel="stylesheet" href="position-fixed.css">
    <title>Posicionamiento fijo</title>
</head>
<body>
    <div id="div1">
        <!-- CONTENIDO -->
    </div>
    <div id="div2">
        <!-- CONTENIDO -->
    </div>
    <footer id="pie-documento">
        Este es el contenido del pie de la página
    </footer>
</body>
</html>
```

Solución

El elemento <footer> debe tener posicionamiento de tipo **fixed** y estar a 0 píxeles de la parte inferior del navegador (**bottom:0px**). El resultado es un contenedor que está «pegado» al extremo inferior de la página independientemente de la posición del resto de elementos.

```
body {
    margin:0px;
}
#div1 {
    width: 100%;
    background-color: rgb(255, 217, 0);
}
#div2 {
    width: 100%;
    background-color:rgb(0, 255, 255);
}
#pie-documento {
    width: 100%;
    height: 100px;
    background-color: rgb(150,150,150);
    position: fixed;
    bottom: 0px;
}
```

© Ediciones Paraninfo

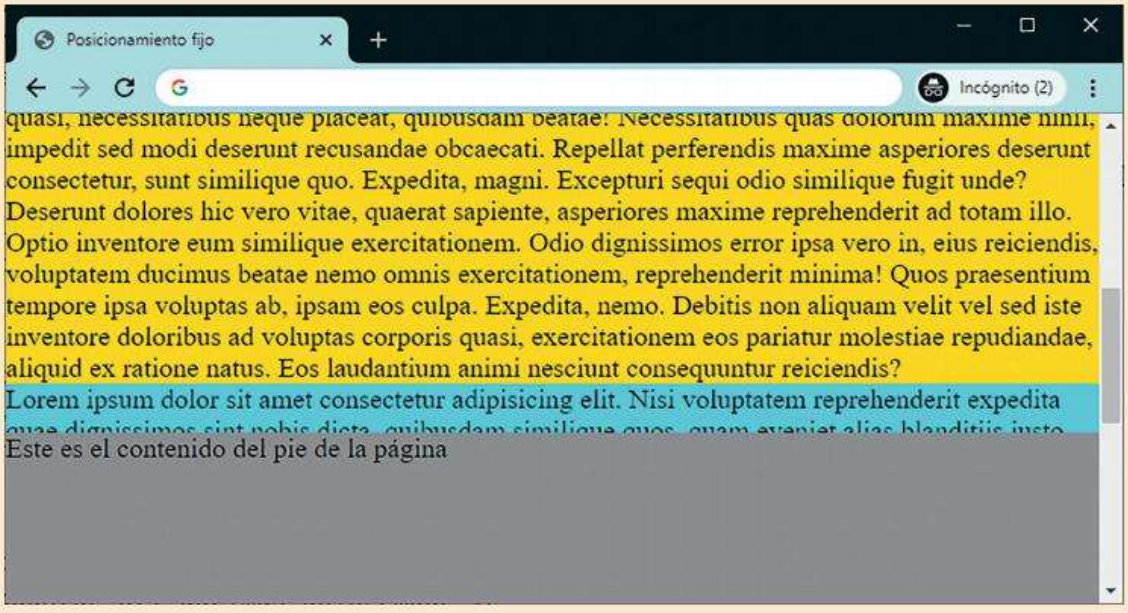


Figura 3.36. *Dando el valor fixed a la propiedad position se puede conseguir fijar la posición de un elemento.*

3.7.4. Propiedad overflow y el contenido desbordante

En ocasiones, el contenido de un contenedor excede el espacio que tiene asignado. Esto se produce porque a un elemento contenedor (por ejemplo, un elemento) se le ha asignado una dimensión y ya no se ajusta al contenido tal y como hace por defecto.

Por defecto, cuando esto sucede el contenido desborda al contenedor. Es decir, sobrepasa sus límites, pero no los modifica, provocando habitualmente un resultado erróneo.

Existen alternativas a este comportamiento por defecto. Se especifican mediante la propiedad **overflow**, que afecta al contenedor si el contenido lo desborda.

Los valores de la propiedad **overflow** se detallan en la Tabla 3.10.

Tabla 3.10. Valores admitidos por la propiedad overflow

Valor	Descripción
visible	Es el valor por defecto. Provoca que el contenido desborde el espacio del contenedor consiguiendo así que sea visible por completo.
hidden	Oculta el contenido que desborda al contenedor.
scroll	Agrega unas barras de desplazamiento *(scroll)* permitiendo así consultar todo el contenido sin que este exceda de los límites del contenedor.
auto	Se comporta igual que el valor **scroll** pero solo si el contenido excede del tamaño del contenedor.

© Ediciones Paraninfo

Por último, señalar que existen propiedades específicas para determinar el comportamiento del contenido desbordante en los ejes x e y. Son las propiedades **overflow-x** y **overflow-y**, y se pueden utilizar conjuntamente.

Actividad propuesta 3.4

Posicionamiento de elementos

Crea el documento HTML y la hoja de estilos CSS que genere una representación lo más parecida a la captura de la Figura 3.37. Presta atención a los siguientes detalles:

- El bloque del lado derecho está alineado a la derecha.
- El texto del bloque central está alineado a la derecha.
- Existe un espacio en blanco entre los bloques superiores y los límites del navegador.
- La línea que delimita los bloques está punteada.
- El pie está fijo en la parte inferior de la página.

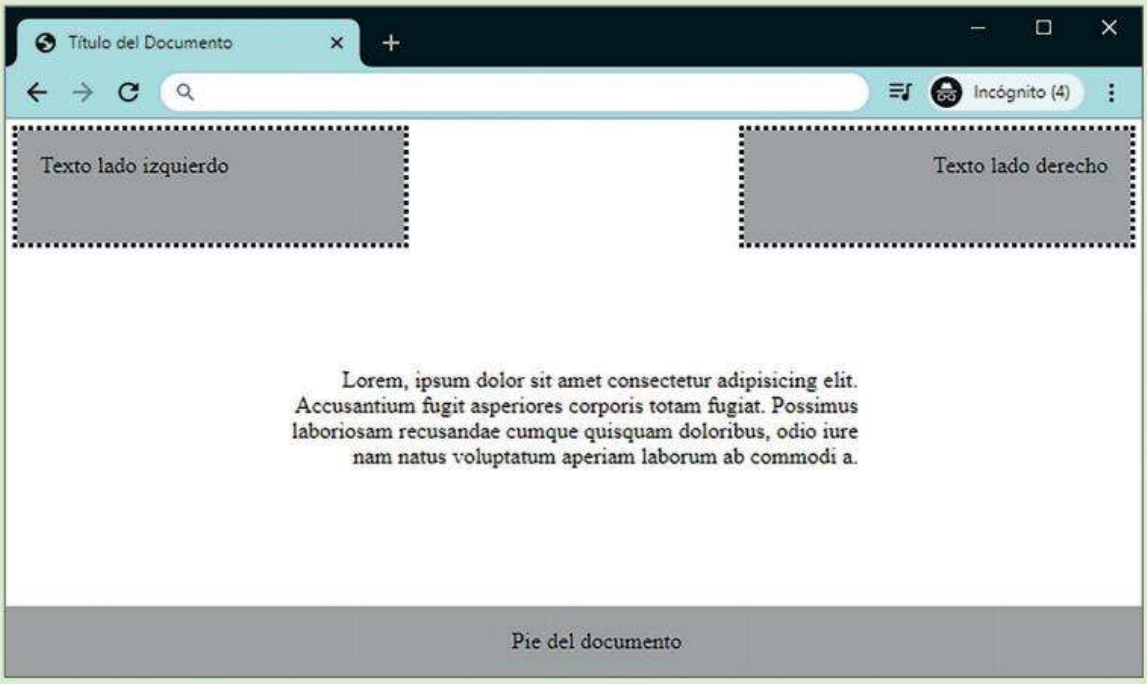


Figura 3.37. *Con las propiedades float y position se puede distribuir el contenido fácilmente por la página.*

3.7.5. Propiedades de texto

Los documentos HTML suelen estar formados, principalmente, por textos. La noticia de un periódico, las características técnicas de un producto o los ingredientes de una pizza son textos y estos deben estar presentados de forma correcta.

Las propiedades de texto permiten especificar prácticamente cualquier característica relacionada con los textos, desde el tipo de fuente hasta el espaciado entre palabras.

En la Tabla 3.11 se detallan las propiedades más importantes, su descripción e indicaciones sobre los valores que pueden tomar.

© Ediciones Paraninfo

Tabla 3.11. Propiedades aplicables a textos

Propiedad	Descripción	Valores
color	Determina el color del texto.	Cualquier valor válido aceptado como color.
font-family	Determina el tipo de letra que se va a utilizar. Si no puede usar el indicado, aplica el tipo de letra por defecto.	Cualquier valor válido aceptado como tipo de letra.
font-size	Determina el tamaño del texto.	*medium, xx-small, x-small, small, large, x-large, xx-large, smaller, larger,* un tamaño indicado en alguna de las unidades aceptadas y un porcentaje sobre el tamaño del texto del contenedor.
font-style	Determina el estilo del texto.	*normal, italic* y *oblique.*
font-weight	Determina el grosor del trazo de los textos.	*normal, bold, bolder, lighter, 100, 200, 300, 400, 500, 600, 700, 800* y *900.*
text-transform	Transforma el texto.	*capitalize, uppercase* y *lowercase.*
text-decoration	Agrega una decoración al texto en forma de línea.	Un valor de las propiedades *text-decoration-line, text-decoration-style* y *text-decoration-color.*
text-decoration-line	Dibuja una línea horizontal sobre el texto.	*underline, overline* y *line-through.*
text-decoration-style	Determina el tipo de línea (solo en el caso de que haya una propiedad *text-decoration-line*).	*none, underline, overline, line-throught* y combinaciones.
text-decoration-color	Determina el color de la línea (solo en el caso de que haya una propiedad *text-decoration-line*).	Un color válido.
text-shadow	Añade una sombra al texto.	Un color válido y una distancia en los ejes x e y indicada en cualquiera de las unidades válidas en CSS.
text-align	Determina la alineación horizontal del texto.	*left, right, center* y *justify.*
line-height	Determina la altura de la línea.	*normal,* un número entero sin unidades, un número expresado en cualquiera de las unidades válidas en CSS.
letter-spacing	Determina el espacio entre las letras.	*normal,* un número expresado en cualquiera de las unidades válidas en CSS.
word-spacing	Determina el espacio entre las palabras.	*normal,* un número expresado en cualquiera de las unidades válidas en CSS.

© Ediciones Paraninfo

3.7.6. Propiedades de listas

Las listas son uno de los elementos de uso más habitual en HTML. Es interesante, por lo tanto, conocer las propiedades más importantes (Tabla 3.12).

Tabla 3.12. Propiedades aplicables a listas

Propiedad	Descripción	Valores más frecuentes
list-style	Permite definir las propiedades *list-style-type, list-style-position* y *list-style-image* en una única línea.	Los valores de las propiedades separados por espacios.
list-style-type	Permite definir la apariencia de la lista en lo referente a la viñeta que precede a cada elemento.	*disc, circle, square, decimal, lower-roman, upper-roman, lower-greek, lower-latin, upper-latin* y *none.*
list-style-position	Determina dónde se colocarán las viñetas respecto a la posición de la lista.	*inside* y *outside.*
list-style-image	Asigna una imagen almacenada en una URL como viñeta.	url("*url-de-la-imagen*");

3.7.7. Propiedades de tablas

Al igual que las listas, las tablas son elementos que se utilizan frecuentemente en la maquetación de páginas HTML. Sus propiedades se recogen en la Tabla 3.13.

Tabla 3.13. Propiedades aplicables a tablas

Propiedades	Descripción	Algunos valores posibles
border-collapse	Determina si los bordes de las celdas de la tabla están fusionados o no.	*collapse* y *separate.*
border-spacing	Determina la distancia entre bordes adyacentes.	Distancia utilizando las unidades de CSS.
caption-side	Determina dónde aparecerá el texto contenido en el elemento <caption>.	*top* y *bottom.*
empty-cells	Indica si las celdas vacías deben mostrarse o no.	*show* y *hide.*
table-layout	Determina el algoritmo que utilizar para diseñar celdas, filas y columnas.	*auto* y *fixed.*
vertical-align	Determina la alineación vertical del contenido de las celdas de una tabla.	*middle, top* y *bottom.*

© Ediciones Paraninfo

3.7.8. CSS aplicado a formularios

La gran mayoría de los estilos existentes se pueden aplicar a los formularios, por lo que a nivel teórico la aportación de esta sección es casi anecdótica. Desde un punto de vista práctico, diseñar un formulario visualmente atractivo requiere utilizar tipos de letras, contenedores que contengan y ubiquen los elementos, márgenes internos y externos para espaciar contenidos, contornos o estados de los elementos. Para ilustrar algunas de estas técnicas, en la Actividad resuelta 3.3 se aplican, por medio de CSS, varias soluciones de diseño sencillas a la vez que efectivas.

Actividad resuelta 3.3

Aplicación de estilo a un formulario de registro

Crear un formulario que cumpla los siguientes requisitos:

- El formulario aparecerá centrado en la pantalla.
- Se utiliza un tipo de letra diferente del que aparece por defecto.
- Se solicitan los siguientes campos:
  - Nombre. Tipo texto.
  - Correo electrónico. Tipo email.
  - Contraseña y confirmación de contraseña. Tipo contraseña.
- Los campos tendrán la etiqueta descriptiva justo encima.
- Al poner el foco sobre un campo de entrada se cambiará el color del borde.
- Los campos de entrada y el botón de aceptación tendrán las esquinas redondeadas.
- El botón de aceptación estará coloreado.

El código HTML correspondiente al formulario es:

```
<div id="div-formulario">
    <form action="registro.php" method="post">
        <h1>Crear cuenta</h1>
        <div>
            <label for="nombre">Nombre</label>
            <input type="text" id="nombre" name="nombre" class="entrada borde-
redondo">
        </div>
        <div>
            <label for="correo">Correo electrónico</label>
            <input type="email" id="correo" name="correo" class="entrada borde-
redondo">
        </div>
        <div>
            <label for="contrasenya">Contraseña</label>
            <input type="password" id="contrasenya" name="contrasenya"
class="entrada borde-redondo">
        </div>
        <div>
            <label for="contrasenya-rep">Confirma tu contraseña</label>
            <input type="password" id="contrasenya-rep" name="contrasenya-rep"
class="entrada borde-redondo" <>
        </div>
```


© Ediciones Paraninfo

```
        <div>
            <input type="submit" value="Crear tu cuenta" id="boton-aceptar"
class="borde-redondo">
        </div>
    </form>
</div>
```

Solución

```
@import url('https://fonts.googleapis.com/css2?family=Roboto');

body {
    font-family: 'Roboto', sans-serif;
}
h1 {
    margin-top:5px;
}
label {
    display: block;
    font-weight: bold;
    margin-bottom: 0.2em;
}
#div-formulario {
    width:350px;
    margin:auto;
}
#boton-aceptar {
    margin-top: 30px;
    width: 100%;
    padding: 12px 3px 12px 3px;
    background-color:  #ffd904;
}
.entrada {
    margin:0px;
    width: 97%;
    padding: 7px 3px 7px 3px;
    margin-bottom: 1em;
}
.entrada:focus {
    outline: none;
    border-color: #ca5f07;
}
.borde-redondo {
    border: 1px;
    border-style: solid;
    border-color: #aaaaaa;
    border-radius: 3px;
}
```

Representación obtenida del formulario:

Figura 3.38. *Un buen diseño mejora la usabilidad de las páginas web.*

© Ediciones Paraninfo

3.7.9. CSS aplicado a enlaces

Los enlaces son los elementos que convierten en documento en hipertextual. Gracias a los enlaces se puede navegar por la web, saltando de recurso en recurso.

El texto de los enlaces, por defecto, se muestra subrayado (utiliza de manera interna la propiedad *text-decoration*) y en color azul cuando no ha sido empleado para navegar a la URL de destino. En el momento de pulsarlo, el color por defecto es rojo. Una vez utilizado, el texto se muestra en color púrpura.

Hay un grupo de cambios de estilo sobre los enlaces que son los mismos que se pueden realizar sobre un texto: el tamaño del texto o el tipo de letra, por ejemplo.

Los cambios específicos referentes al enlace se realizan sobre los diferentes estados por los que este puede pasar. Mediante selectores que incluyan pseudoclases se puede hacer referencia a dichos estados para asignar cambios en la apariencia del enlace.

Tabla 3.14. Selectores relativos a los estados por los que puede pasar un enlace

Selector	Estado
a:link	Es el estado por defecto de un enlace.
a:visited	Se activa cuando el enlace ha sido activado. Suele estar deshabilitado en los navegadores modernos por razones de seguridad.
a:hover	Se activa cuando se pasa el ratón por encima del enlace.
a:focus	Se activa cuando el foco se posiciona en el enlace.
a:active	Se activa en el momento de la activación del enlace.

3.7.10. Fondos, bordes y contornos

Existe un importante número de propiedades para caracterizar el fondo, los bordes y los contornos de los elementos. Se pueden aplicar a todo tipo de elementos: ítems de una lista (<li>), celdas de una tabla (<td>), contenedores de bloque (, <footer>, <article>…), contendedores de línea (), etcétera.

Las propiedades relativas al fondo comienzan por el prefijo **background** y permiten determinar un color o una imagen. En caso de ser una imagen, existen un buen número de propiedades con características relativas a cómo se deben presentar.

Las propiedades relativas al borde de los elementos comienzan por el prefijo **border** y permiten configurar el color, el tipo, el grosor, redondear las esquinas y un largo etcétera.

Al igual que los bordes, se pueden definir contornos (cuya diferencia fundamental es que estos no ocupan espacio) utilizando las propiedades que comienzan por **outline.**

© Ediciones Paraninfo

Referencia

Si deseas profundizar más en los atributos relativos al fondo, los bordes y los contornos, puedes hacerlo en los siguientes enlaces:

https://developer.mozilla.org/es/docs/Web/CSS/CSS_Background_and_Borders

https://developer.mozilla.org/es/docs/Web/CSS/outline

3.7.11. Diseño adaptativo con *media queries*

Con anterioridad a la llegada de los teléfonos móviles inteligentes y de las tabletas, el ordenador era prácticamente el único dispositivo utilizado para navegar por la web. Aunque los tamaños y las proporciones de las pantallas son variados, no resultaba difícil crear un diseño de página HTML capaz de presentarse de forma correcta en dichas pantallas.

En la actualidad la diversidad de tamaños y proporciones de dispositivos es mucho mayor: pequeñas pantallas de teléfono móvil y grandes pantallas de ordenador son medios habituales para acceder a las páginas web.

Esta diversidad requiere de técnicas flexibles que permitan adaptar las reglas de la maquetación con CSS al tamaño o distribución del dispositivo. A estas técnicas se les conoce con el nombre de **diseño adaptativo** o ***responsive.***

El diseño adaptativo se puede conseguir utilizando *media queries,* que es un mecanismo para asignar propiedades y valores CSS distintos en función del tipo de dispositivo (pantallas, impresoras o sintetizadores de voz) y de sus características, principalmente el tamaño.

Las *media queries* son, en resumen, un conjunto de reglas que determinan qué valores asignar a determinadas propiedades. El estándar es complejo, pero se puede utilizar a un nivel sencillo y obtener unos resultados más que interesantes.

La manera más sencilla de aplicar las *media queries* es a través del elemento <link>, asignando diferentes hojas de estilo en función del tamaño de la página.

En el siguiente código se muestra un ejemplo en el que se asignan una u otra hoja de estilo CSS en función del ancho del medio en el que se está visualizando el documento: se aplica estilo1.css si el ancho máximo de 1000 píxeles, o estilo2.css si el ancho del medio tiene un mínimo de 1001 píxeles.

```
<link rel="stylesheet" href="estilo1.css" media="(max-width:1000px)">
<link rel="stylesheet" href="estilo2.css" media="(min-width:1001px)">
```

Esta solución puede resultar fácil y rápida, pero supone tener duplicada (al menos) la hoja de estilos y esto no suele ser una buena decisión. Para evitarlo existe la alternativa de definir reglas en función del medio que afecten a una selección.

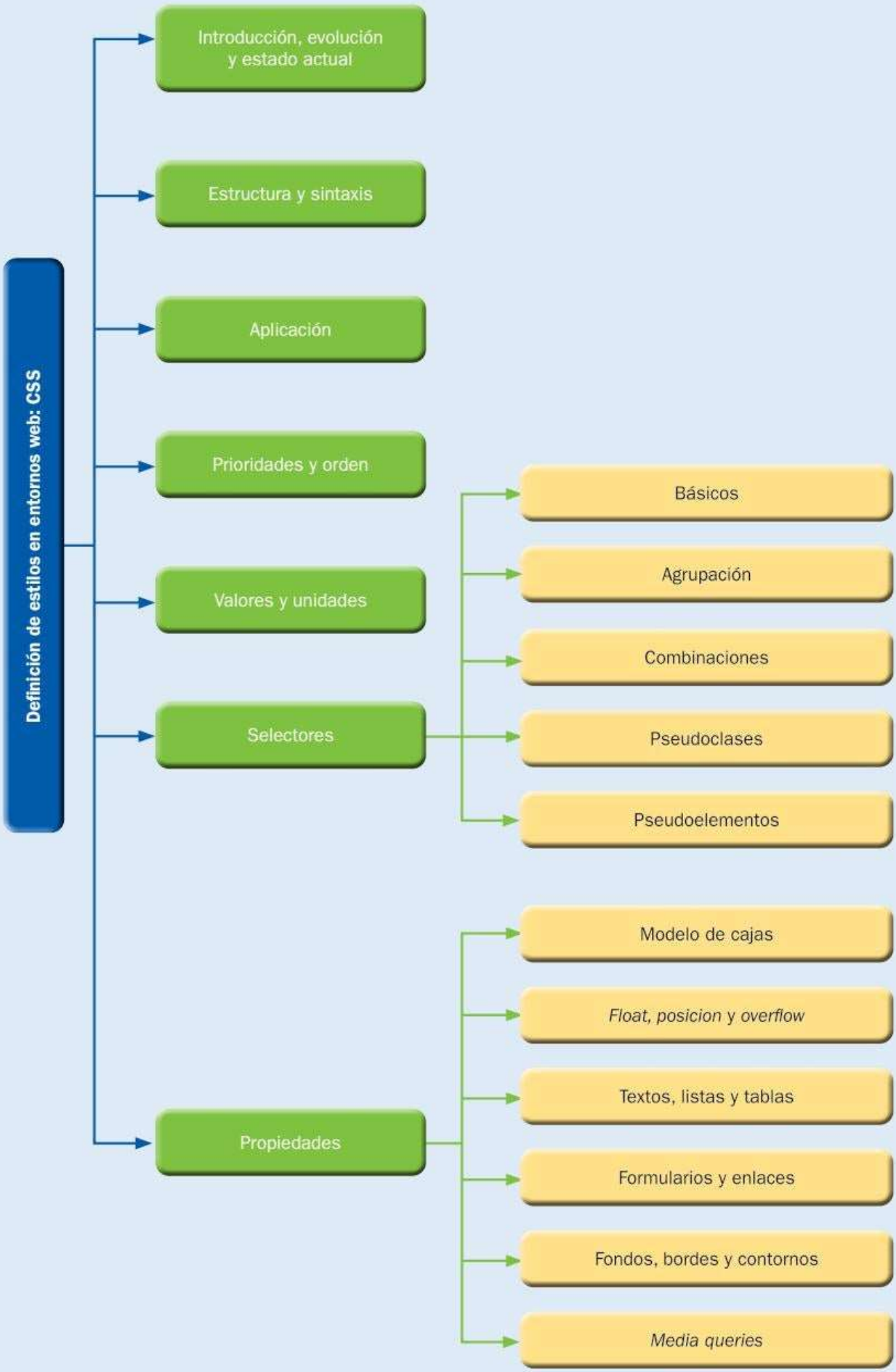
© Ediciones Paraninfo

```
@media (max-width:1000px) {
    #parrafo {
        color:springgreen;
    }
}
@media (min-width:1001px) {
    #parrafo {
        color:rgb(171, 15, 199);
    }
}
```

Los parámetros que pueden participar de las *media queries* son muchos y no se van a detallar en este libro, pero es conveniente saber que dichos parámetros pueden hacer referencia al tipo de dispositivo (pantalla, impresora, sintetizadores de voz), el número de colores que soporta el dispositivo, las ratios o proporciones de aspecto, el alto y ancho del dispositivo, la orientación o la resolución. Además, dispone de la posibilidad de incluir operadores lógicos para crear condiciones más complejas.

© Ediciones Paraninfo

MAPA CONCEPTUAL



© Ediciones Paraninfo

Actividades de comprobación

3.1. CSS se puede aplicar a un documento HTML:

a) Como un documento externo al documento.
b) Como una declaración interna dentro del documento.
c) Como una asignación *inline* (dentro del mismo elemento).
d) Todas las respuestas son correctas.

3.2. ¿Qué factor o factores participan en la decisión de qué estilo aplicar en CSS?

a) Origen e importancia.
b) Nivel de especificidad.
c) Orden de aparición.
d) Todas las respuestas son correctas.

3.3. Indica cuál de las siguientes reglas se impondría al resto en CSS:

a) La que utiliza el selector universal (*).
b) La que utiliza un selector de elemento.
c) La que utiliza un selector de clase.
d) La que utiliza un selector de identificador.

3.4. ¿Cómo se representa en CSS la unidad píxel?

a) pxl
b) px
c) pxlx
d) pix

3.5. La unidad relativa em ¿con qué tiene relación?

a) Con el tamaño de letra del elemento padre.
b) Con el tamaño de letra del elemento hijo.
c) Con el tamaño de letra del navegador.
d) Con el tamaño de letra de la página principal.

3.6. ¿Qué significan las siglas RGB en los códigos de colores?

a) *Red, Grey, Blue.*
b) *Red, Green, Black.*
c) *Red, Green, Blue.*
d) *Red, Grey, Black.*

3.7. Indica qué selector usarías si quieres aplicar una regla a todos los elementos de una página:

a) p
b) div
c) *
d) +

© Ediciones Paraninfo

3.8. ¿Cuál de los siguientes selectores significa que <p> debe ser hijo de ?

a) p + div
b) p div
c) p > div
d) p * div

3.9. ¿Cuál de las siguientes pseudoclases significa que el elemento no debe tener hijos?

a) div:default
b) div:empty
c) div:focus
d) div:disabled

3.10. ¿Qué selector utilizarías si quieres asignar el mismo estilo a los elementos h1 y h2?

a) h1>h2
b) h1+h2
c) h1,h2
d) h1*h2

3.11. ¿Cuál es la propiedad que crea un espacio en blanco en el interior del componente?

a) margin.
b) padding.
c) border.
d) outline.

3.12. ¿Cuál es la propiedad que crea un espacio en blanco en el exterior del componente?

a) margin.
b) padding.
c) border.
d) outline.

3.13. Por defecto, los elementos de bloque:

a) Crean un salto de línea después del elemento.
b) Ocupan el 100 % del espacio disponible.
c) Se les aplican los valores de las propieades width y height.
d) Todas las afirmaciones son correctas.

3.14. ¿Qué valor hay que asignar a la propiedad float para que un elemento «flote» hacia el lado derecho?

a) left.
b) right.
c) inherit.
d) none.

3.15. ¿Qué valor de la propiedad position fija una coordenada absoluta respecto a la página completa?

a) static.
b) relative.
c) absolute.
d) fixed.

© Ediciones Paraninfo

3.16. ¿Qué ocurre si se asigna el valor auto al atributo overflow y el contenido excede del tamaño disponible?

a) El contenido desborda al contenedor haciendo que sea completamente visible.
b) El contenido que excede del espacio disponible se oculta.
c) Se muestran unas barras de desplazamiento sean o no necesarias.
d) Se muestran unas barras de desplazamiento si son necesarias.

3.17. ¿Con qué atributo se puede convertir a mayúsculas todos los caracteres de un texto?

a) font-style.
b) font-size.
c) text-transform.
d) font-weight.

Actividades de aplicación

3.18. Crea una hoja de estilos CSS y asígnasela a un documento HTML. Comprueba que se relaciona correctamente.

3.19. Crea un estilo *inline* asociado a un elemento <p> que muestre el texto en cursiva. Comprueba que el resultado es correcto.

3.20. En un documento HTML crea un elemento . Haz, mediante CSS, que el elemento mida el 50 % del ancho del espacio de la ventana del navegador y 200 píxeles de alto.

3.21. En un documento HTML crea un elemento . Haz, mediante CSS, que el elemento mida 200 píxeles de ancho por 200 píxeles de alto. Asígnale el color de fondo #883344.

3.22. En un documento HTML crea un elemento . Haz, mediante CSS, que el elemento mida el 100 % del ancho del espacio de la ventana del navegador por 200 píxeles de alto. Asígnale una imagen de fondo.

3.23. Crea un documento HTML que esté formado por cinco elementos . Haz que los impares se muestren con un color de fondo distinto de los pares.

3.24. Crea un documento HTML que contenga una tabla. Haz que la primera y la última fila muestren el texto en negrita.

3.25. En un documento HTML crea un elemento . Haz, mediante CSS, que el elemento mida 50 píxeles de ancho por 50 píxeles de alto. Asígnale un color de fondo #338899. Haz que el aparezca centrado.

3.26. En un documento HTML crea un elemento . Haz, mediante CSS, que el elemento mida 50 píxeles de ancho por 50 píxeles de alto. Consigue que muestre un borde de 3 píxeles de ancho de color rojo.

© Ediciones Paraninfo

3.27. Utilizando el modo de display grid, crea una estructura de rejilla de 3 × 3 elementos .

3.28. Crea un pie de página que ocupe el 100 % del ancho de la ventana del navegador y que tenga una transparencia del 50 %.

3.29. Crea una tabla y, mediante CSS, consigue que el texto de las columnas primera y última aparezca centrado y en letra negrita.

Actividades de ampliación

3.30. Busca ejemplos de CSS aplicado a tablas y estudia las reglas que utilizan.

3.31. Encuentra información sobre las transformaciones CSS.

3.32. Busca información sobre Bootstrap e infórmate sobre su utilidad.

3.33. En la web hay multitud de recursos; halla generadores de gradientes CSS.

3.34. Accede al formulario de inscripción de tu red social favorita y analiza su diseño CSS.

3.35. Busca plantillas (templates) CSS en la web. Crea en tu navegador una sección de favoritos que consideres que te van a hacer falta en el futuro. No dejes de ampliarlo con nuevos enlaces.

Enlaces web de interés

- **CSS w3c.es** - https://www.w3.org/Style/CSS/
  (Información oficial en inglés sobre CSS)
- **MDN Web Docs - CSS** - https://developer.mozilla.org/es/docs/Web/CSS
  (Portal web de Mozilla para la documentación de los estándares web)
- **w3schools** - https://www.w3schools.com/css/
  (Tutorial de CSS de w3schools)
- **CSS3 Generator** - https://css3generator.com/
  (Generador de código CSS3)

© Ediciones Paraninfo

UNIDAD 4

XML: creación, validación y utilización

Objetivos

- Conocer el origen de XML y descubrir su utilidad.
- Aprender cómo construir un documento XML estructural y sintácticamente correcto.
- Enumerar las alternativas existentes para validar documentos XML, detallando las reglas y posibilidades de cada una de ellas.
- Exponer las aplicaciones prácticas del lenguaje XML en diferentes ámbitos.
- Presentar algunas de las herramientas existentes para la consulta y edición de documentos XML.

Contenidos

- 4.1. Introducción, evolución y estado actual
- 4.2. Estructura y sintaxis de XML
- 4.3. Validación de XML
- 4.4. XML aplicado
- 4.5. Herramientas de visualización y edición de XML

Introducción

XML es, junto con HTML, uno de los lenguajes de marcas más característicos. En realidad, XML es un metalenguaje y la base para la construcción de otros lenguajes más específicos.

XML define las reglas de construcción de los documentos, pero no se especifican las etiquetas tal y como se hace en HTML. En XML, en lugar de haber elementos o atributos predefinidos, hay mecanismos para definirlos.

Al tratarse de un estándar, alrededor de XML hay un buen número de tecnologías y herramientas: mecanismos de validación, editores, programas de visualización o librerías de programación para integrarlo en desarrollos de software propios.

4.1. Introducción, evolución y estado actual

Cuando se habla de lenguajes de marcas, los dos ejemplos más citados son HTML y XML. HTML es, como se explica en este mismo libro, el lenguaje de la web: un conjunto de etiquetas comprensibles e interpretables por el navegador que posibilitan representar cualquier información para que la puedan consultar los usuarios.

El objetivo de XML es, en cambio, menos concreto. No está orientado a la web, pero puede utilizarse para presentar datos en un navegador como HTML. Tampoco está orientado a la representación de interfaces gráficas de usuario, pero se utiliza en un buen número de tecnologías para ello. También se puede utilizar para intercambiar información entre sistemas, guardar la configuración de una aplicación o difundir información mediante RSS.

El lenguaje de marcado XML (*eXtensible Markup Language,* «lenguaje de marcas extensible», en castellano) es, en realidad, un metalenguaje de propósito general desarrollado por el W3C (World Wide Consortium) y se utiliza en multitud de contextos.

Tabla 4.1. Evolución de las versiones de XML

Año	Versión	Observaciones
1998	XML 1.0	Primera edición
2000	XML 1.0	Segunda edición
2004	XML 1.0	Tercera edición
2004	XML 1.1	Primera edición
2006	XML 1.0	Cuarta edición
2006	XML 1.1	**Segunda edición**
2008	XML 1.0	**Quinta edición**

© Ediciones Paraninfo

XML proviene del lenguaje GML *(Generalized Markup Language)* y de SGML *(Standard Generalized Markup Language).* Fue **lanzado en 1998** y se encuentra en la **versión 1.1.** Es un **formato abierto,** lo que implica que se puede utilizar de manera libre y sin coste. La extensión de los ficheros XML es **.xml** y el tipo MIME es **application/text** y **text/xml.**

El W3C anima a crear los documentos en XML 1.0 salvo que se necesiten las nuevas funciones añadidas en XML 1.1. En la gran mayoría de los casos estas funciones no son necesarias, por lo que habitualmente los documentos XML se encuentran en versión 1.0.

A continuación, se muestra un ejemplo de un documento XML 1.0:

```
<?xml version="1.0" encoding="UTF-8"?>
<animal>
  <nombre>Tiburón blanco</nombre>
  <nombrecientifico>Carcharodon carcharias</nombrecientifico>
  <familia>Lamnidae</familia>
  <estadodeconservacion>Vulnerable</estadodeconservacion>
</animal>
```

Sin entrar en detalles (se tratarán en el siguiente apartado), se puede observar que el formato es sencillo: texto plano, etiquetas de apertura y cierre o estructura jerárquica son algunas de las características de este tipo de documentos.

XML tiene fundamentos robustos y una sintaxis muy simple, lo que lo convierte en una herramienta muy versátil y que goza de una gran aceptación. XML y sus dialectos se utilizan en múltiples contextos; creación de ficheros de configuración, diseño de interfaces gráficas, publicación de contenidos, intercambio de información entre aplicaciones o diseño de páginas web son solo algunas de las áreas de aplicación en las que se emplea.

A pesar de la sencillez de XML, existe un importante conjunto de tecnologías alrededor que convierten el lenguaje es una herramienta poderosa: validadores, transformadores, editores… La estandarización a la que se somete permite que el ecosistema de aplicaciones relacionadas con el lenguaje sea uno de sus activos más importantes.

Nota técnica

Un metalenguaje es una base para crear lenguajes. No tiene etiquetas concretas ni un ámbito de aplicación definido. Un metalenguaje contiene reglas para ser la base de la creación de lenguajes robustos. XHTML, por ejemplo, es un lenguaje basado en el metalenguaje XML.

4.2. Estructura y sintaxis de XML

Los documentos XML se estructuran en dos partes principales: **prólogo** y **cuerpo.** El prólogo contiene información relativa al conjunto del documento y el cuerpo recoge los elementos con la información propiamente dicha. También permite incluir comentarios que no serán procesados.

© Ediciones Paraninfo

4.2.1. Declaración

La declaración del documento XML se realiza en el prólogo, siendo este opcional. Si existe, debe ser la primera línea del documento. En él se indica que el documento es XML, la versión y la codificación utilizada mediante los siguientes atributos:

- versión. Versión de XML.
- encoding. Codificación utilizada en el documento.
- standalone. Indica si el documento es independiente o si existe un DTD externo para realizar la validación. Admite los valores **yes** (es independiente) y **no** (está asociado a un DTD de validación). Este atributo es opcional.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
```

Nota técnica

Un DTD *(Document Type Definition)* contiene reglas que determinan que un documento XML es válido en un contexto determinado. En el Apartado 4.3. se explica en detalle qué son y cómo se construyen las validaciones.

4.2.2. Estructura, sintaxis, elementos y atributos

El cuerpo del documento XML está compuesto por elementos. Los elementos son el componente fundamental de XML. Se parecen mucho a los elementos de HTML, pues tienen una etiqueta de apertura y otra de cierre, que en el caso de XML es obligatoria.

Las etiquetas de apertura tienen un nombre y, opcionalmente, uno o más atributos separados por espacios. La etiqueta de cierre debe tener el mismo nombre que la de apertura, pero este estará precedido por el símbolo /.

```
<nombre-elemento nombre-atributo-1="valor1" nombre-atributo-2="valor2" >
</nombre-elemento>
```

Las reglas para la asignación de nombres de elementos son las siguientes:

- Diferencian entre mayúsculas y minúsculas.
- Deben comenzar con una letra o un guion bajo.
- Los nombres de elementos tienen que ser idénticos en las etiquetas de apertura y de cierre.
- Los nombres pueden estar formados por caracteres alfanuméricos, guiones, guiones bajos y puntos.
- Los nombres no pueden contener espacios (interpreta como nombre del elemento la primera palabra y el resto como parámetros).

© Ediciones Paraninfo

Por ejemplo, el siguiente elemento generará un error al validarlo con un navegador web:

```
<1color>Azul</1color>
```

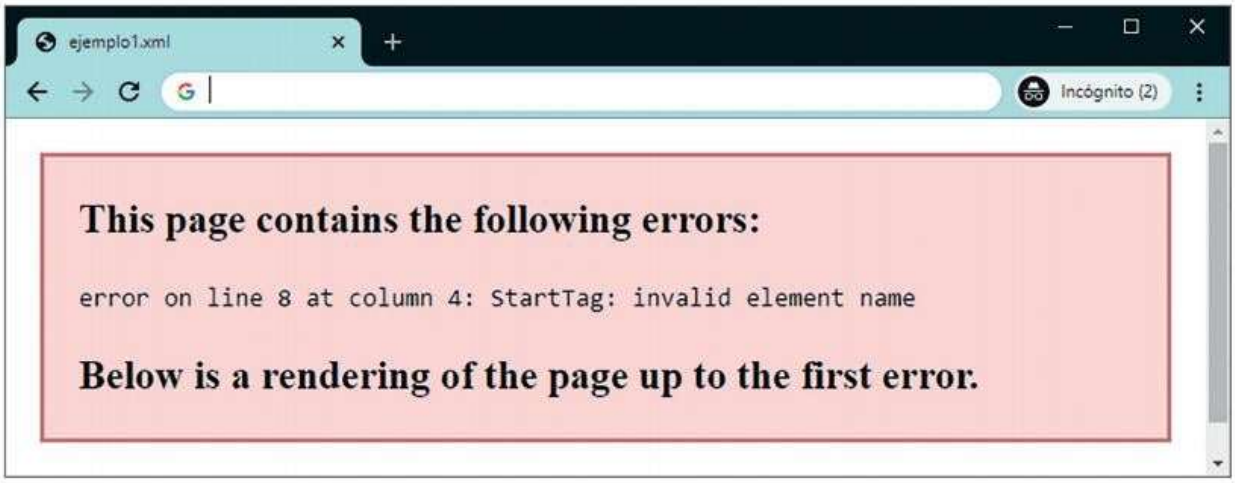


Figura 4.1. *Con XML se garantiza que los documentos estén bien formados.*

También serán erróneos los siguientes elementos:

```
<color>Azul</Color>
```

```
<-color>Azul</-color>
```

```
<.color>Azul</.color>
```

```
<el color>Azul</el color>
```

En cambio, los siguientes elementos serán procesados correctamente:

```
<color1>Azul</color1>
```

```
<_color>Azul</_color>
```

```
<COLOR>Azul</COLOR>
```

```
<el_color>Azul</el_color>
```

Las reglas de creación de atributos se muestran a continuación:

- Deben tener asignado un valor.
- Los valores siempre van entrecomillados, admitiendo comillas simples y dobles (deben coincidir el tipo de comilla de cierre con el de apertura).
- Diferencian entre mayúsculas y minúsculas.

© Ediciones Paraninfo

- Deben comenzar con una letra o un guion bajo.
- Pueden estar formados por caracteres alfanuméricos, guiones, guiones bajos y puntos.

Ejemplo de atributo válido:

```
<nombre familia="Lamnidae">Tiburón blanco</nombre>
```

En XML se puede caracterizar un elemento mediante atributos o elementos hijos. No existe una regla para determinar en qué caso se deben utilizar uno u otro, pero sí casuísticas en la que los atributos no son suficientes para crear la estructura de datos que se desea.

Los elementos pueden contener texto u otros elementos. En el segundo caso, la relación entre los elementos es jerárquica: un elemento puede contener a otros elementos y estos, a su vez, ser contenedores de otros más. Esta estructura se denomina «de árbol» y se caracteriza por tener un único elemento en la parte alta de la jerarquía (denominado *raíz*) que se ramifica hasta llegar a los nodos finales u hojas.

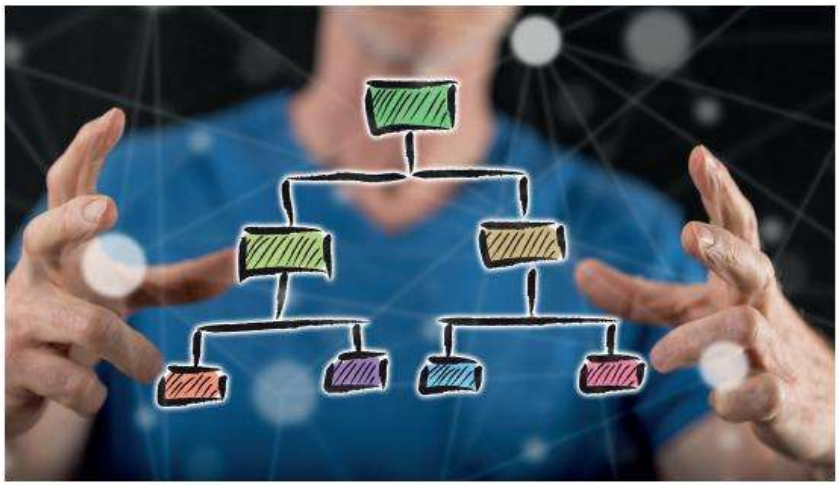


Figura 4.2. *Los documentos XML son estructuras jerárquicas en forma de árbol.*

Las relaciones entre los elementos de un documento XML se denominan de la siguiente manera:

- Existe un único elemento raíz (**root**).
- Cada uno de los elementos descendientes directos de un elemento se llama *hijo* (**children**).
- El elemento ascendiente directo de un elemento se llama *padre* (**parent**).
- Los elementos que tienen un padre común se denominan *hermanos* (**sibling**).

En el ejemplo que se muestra a continuación las relaciones entre los elementos son las siguientes:

- **<animales>** es el elemento raíz.
- Los elementos **<animal>** son hijos (**children**) del elemento **<animales>**.
- El elemento **<animales>** es padre (**parent**) de los elementos **<animal>**.

© Ediciones Paraninfo

- Los elementos **<animal>** son hermanos (**sibling**) entre sí.
- Los elementos **<nombre>** y **<nombrecientifico>** son hijos del elemento **<animal>**.

```
<?xml version="1.0" encoding="UTF-8"?>
<animales>
  <animal id="1">
    <nombre>Tiburón blanco</nombre>
    <nombrecientifico>Carcharodon carcharias</nombrecientifico>
  </animal>
  <animal id="2">
    <nombre>Atún rojo</nombre>
    <nombrecientifico>Thunnus thynnus</nombrecientifico>
extinción</estadodeconservacion>
  </animal>
</animales>
```

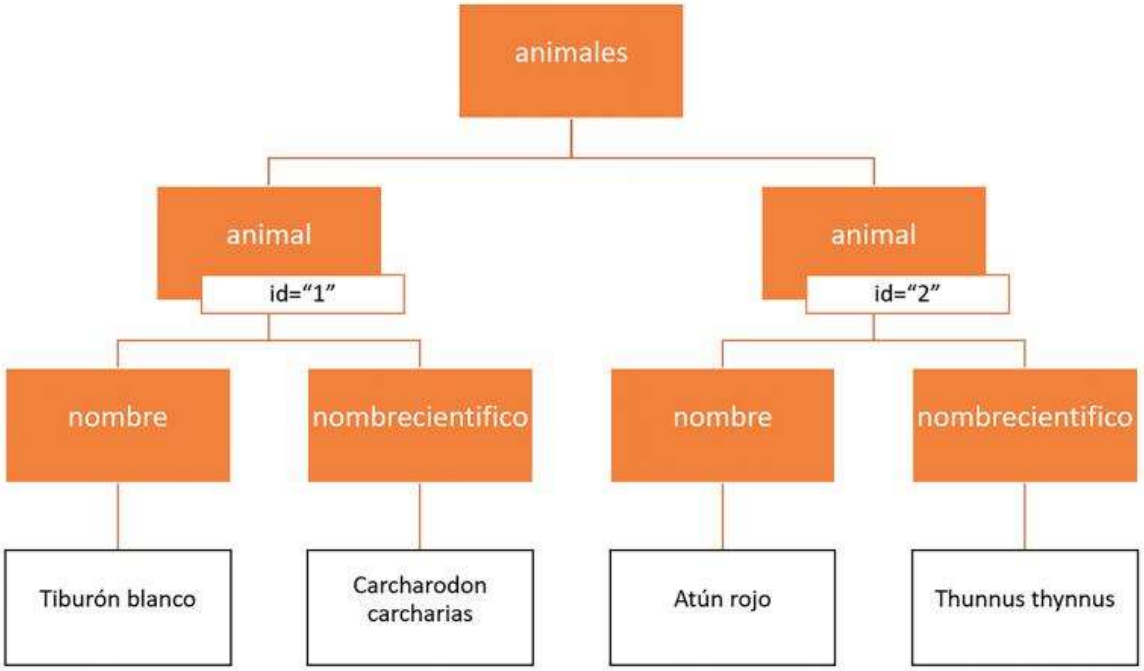


Figura 4.3. *Las relaciones entre los elementos de un documento XML se asimilan con relaciones parentales.*

Actividad resuelta 4.1

Representación de un *curriculum vitae*

Representar en un documento XML un *curriculum vitae*. Los datos que debe contener son los siguientes:

- Nombre y apellidos.
- Año de nacimiento.
- Residencia:
  - Ciudad.
  - País.

© Ediciones Paraninfo

- Datos de contacto:
  - Teléfono.
  - Correo electrónico.
  - Perfil LinkedIn.
- Breve descripción general.
- Listado de competencias, indicando el nombre y el nivel.
- Formación académica, con la titulación, el centro de formación y el año de titulación.
- Experiencia profesional, especificando el puesto, la empresa, el año de inicio y el año de finalización.

Solución

```
<?xml version="1.0" encoding="UTF-8"?>
<cv>
    <nombre>Shigeru Miyamoto</nombre>
    <anyo-nacimiento>1952</anyo-nacimiento>
    <residencia>
        <ciudad>Kioto</ciudad>
        <pais>Japón</pais>
    </residencia>
    <contacto>
        <telefono>+81-555-11-33-66</telefono>
        <email>shigeru@servidordecorreo.jp</email>
        <linkedin>http://www.linkedin.com/miyamoto</linkedin>
    </contacto>
    <descripcion>
    Gran capacidad creativa. Interesado en el diseño de videojuegos.     </
descripcion>
    <competencias>
        <competencia nombre="Diseño" nivel="Alto"/>
        <competencia nombre="Trabajo en equipo" nivel="Medio"/>
        <competencia nombre="Música" nivel="Alto"/>
    </competencias>
    <formacion>
        <titulacion nombre="Diseño industrial" centro="Colegio Municipal de Artes
Industriales de Kanazawa" anyo="1975"/>
    </formacion>
    <experiencia-profesional>
        <experiencia>
            <puesto>Diseñador de juegos</puesto>
            <empresa>Nitendo</empresa>
            <inicio>1977</inicio>
            <fin></fin>
        </experiencia>
    </experiencia-profesional>
</cv>
```

Actividad propuesta 4.1

Representación de un viaje turístico

Representa en un documento XML un viaje turístico. Los datos que debe contener son los siguientes:

© Ediciones Paraninfo

- Nombre y apellidos de los clientes.
- Fecha de inicio.
- Fecha de fin.
- Origen del viaje.
- Una relación de destinos, cada uno de los cuales contendrá la siguiente información:
  - Nombre de la ciudad.
  - Nombre del hotel.
  - Número de noches.
- Precio del viaje.

4.2.3. Comentarios

XML admite incluir comentarios en el interior de los documentos. Están formados por textos delimitados por los caracteres de apertura **<!--** y de cierre -->.

Los comentarios se escriben para ser leídos por personas, por lo que el contenido no tiene que cumplir con ninguna sintaxis ni regla; contiene un texto en lenguaje natural que explica, aclara o informa de algo relevante para la comprensión del documento por parte de la persona que lo está leyendo. No son procesados, ni validados, ni participan de la estructura de datos que representa el documento.

```
<!-- Este texto es un comentario -->
```

4.2.4. Espacios de nombres

El **espacio de nombres** o **namespace** es un identificador que se utiliza para resolver las ambigüedades que podrían surgir en caso de que haya elementos o atributos con el mismo nombre.

Por ejemplo, dos vocabularios de XML podrían utilizar el elemento <autor>. Si esos dos vocabularios se utilizan en el mismo documento, surge la pregunta siguiente: ¿cómo saber cuando se encuentra un elemento <autor> a qué vocabulario de XML corresponde?

Es para problemas como este cuando el espacio de nombres surge y tiene utilidad, ya que permitiría relacionar cada uno de los elementos <autor> con un «cualificador» distinto (el espacio de nombres) que resolvería de forma inequívoca la confusión al referenciar a dicho elemento.

La declaración de un espacio de nombres se realiza utilizando el atributo reservado **xmlns**, seguido del nombre del espacio de nombres y una URL:

```
xmlns:nombre-espacio-nombres=URL
```

© Ediciones Paraninfo

En el ejemplo referenciado anteriormente, se podrían declarar dos espacios de nombres para los dos vocabularios utilizados:

```
xmlns:autorestecnicos='http://unaurlcualquiera';
xmlns:autoresliterarios='http://otraurlcualquiera';
```

La ambigüedad se resolvería referenciando al elemento <autor> incluyendo su **namespace** de la siguiente manera: <autorestecnicos:autor>.

4.2.5. Entidades

Las entidades XML son un mecanismo para representar información dentro de un documento haciendo referencia a ella en lugar de incluirla directamente. Esta información puede ser desde un carácter predefinido hasta un fichero externo. Gracias a las entidades se puede repartir el contenido de los documentos en varias fuentes.

Esta fragmentación aporta una mejor estructura, facilita el trabajo en grupo y permite la reutilización, ya que las entidades pueden ser referenciadas desde diversos documentos principales. De cara al proceso no tiene consecuencias, pues el programa que procese el XML lo hará sobre el conjunto total de información, como si fuese una única unidad.

Las entidades se definen en el DTD. Al declarar una entidad se asigna un alias a un bloque de texto. Ese alias se puede adjuntar a un documento XML para incrustar el bloque correspondiente.

Por ejemplo, es posible crear una entidad que almacene la ficha de una empresa (nombre, dirección y CIF) e incrustarla en varios documentos sin necesidad de escribir el mismo texto una y otra vez. En caso de que la empresa cambie de dirección, solo habrá que cambiarla en un punto.

Existen categorías principales de entidades en XML: las **entidades generales** y las **entidades de parámetros.** La entidad general es un alias que se convierte en parte del documento XML. La entidad de parámetros se convierte, en cambio, en parte del DTD.

Las entidades generales se dividen, a su vez, en dos tipos: **internas** y **externas.** Las entidades internas mantienen sus valores en la propia declaración de la entidad. Las entidades externas mantienen los valores en un archivo externo.

Existen cinco entidades internas predefinidas de uso común. Representan caracteres que provocarían ambigüedad en el procesado del XML, ya que tienen un significado estructural. Se muestran en la Tabla 4.2.

Tabla 4.2. Entidades predefinidas

Entidad	Significado	Carácter
<	Símbolo de «menor que»	<
>	Símbolo de «mayor que»	>

© Ediciones Paraninfo

Entidad	Significado	Carácter
&	Símbolo del *ampersand* (en español, *et*)	&
'	Comilla simple	‘
"	Comilla doble	“

El siguiente documento XML de ejemplo contiene dos errores:

- El símbolo & en el valor del atributo **nombre** se interpreta como el comienzo de una entidad.
- Las comillas que rodean la palabra **sueño** en el valor del atributo **mensaje** se confunden con las comillas que delimitan la cadena de caracteres.

```
<?xml version="1.0" encoding="UTF-8"?>
<ficha>
    <empresa nombre="Toledo & Kurylenko S.L"></empresa>
    <eslogan mensaje="Vendemos "sueños" imposibles"></eslogan>
</ficha>
```

Al tratar de abrirlo desde un navegador, se detendrá al detectar el primero de los errores (los procesadores de XML de los navegadores suelen parar en el momento en el que encuentran el primer error durante el análisis del documento) (Figura 4.4).

```
Error de lectura XML: mal formado
Número de línea 3, columna 30:
    <empresa nombre="Toledo & Kurylenko S.L"></empresa>
-----------------------------^
```

Figura 4.4. *Los navegadores detectan si el documento XML está mal formado.*

En ambos casos, la solución consiste en sustituir los caracteres problemáticos por las entidades correspondientes:

```
<?xml version="1.0" encoding="UTF-8"?>
<ficha>
    <empresa nombre="Toledo &amp; Kurylenko S.L"></empresa>
    <eslogan mensaje="Vendemos &quot;sueños&quot; imposibles"></eslogan>
</ficha>
```

```
-<ficha>
    <empresa nombre="Toledo & Kurylenko S.L"/>
    <eslogan mensaje="Vendemos "sueños" imposibles"/>
  </ficha>
```

Figura 4.5. *Las entidades permiten representar caracteres problemáticos.*

© Ediciones Paraninfo

4.2.6. CDATA

En XML una sección CDATA contiene un conjunto de caracteres que no debe ser tratado por el analizador. Las secciones CDATA se utilizan habitualmente para almacenar código XML o HTML que, de otra forma, impediría que el documento XML contenedor no estuviese bien formado.

Nota técnica

Se dice que un documento XML está bien formado cuando es conforme a la sintaxis de XML. Este y otros aspectos formales relacionados con la validación de documentos XML se tratan en el Apartado 4.3.

En una sección CDATA no es necesario sustituir los caracteres conflictivos por entidades, ya que son ignorados por el analizador. Si se incluyen etiquetas XML dentro de una sección CDATA, no serán tratadas como tal, sino como una sucesión de caracteres sin ningún sentido específico.

Las secciones CDATA comienzan por la secuencia de caracteres:

```
<![CDATA[
```

Terminan por la secuencia:

```
]]>
```

En el siguiente código de ejemplo se muestra un documento XML que incluye código HTML. Dentro del código HTML hay un elemento <br> que no cumple con las reglas de XML, pues no tiene etiqueta de cierre. Por lo tanto, es necesario incluir el código HTML como una sección CDATA para que el documento XML esté bien formado.

```
<?xml version="1.0" encoding="UTF-8"?>
<ficha>
    <nombre>Lenguaje C</nombre>
    <info-html>
        <![CDATA[
            <h1>LENGUAJES DE PROGRAMACIÓN<h1>
            <hr>
            <a href="url-enlace">C</a>
        ]]>
    </info-html>
</ficha>
```

Como se puede suponer, la información contenida en secciones CDATA no forma parte de la estructura jerárquica del documento, aunque se trate de elementos XML. El elemento CDATA se tratará como tal, considerándose una única unidad de la estructura.

© Ediciones Paraninfo

4.3. Validación de XML

XML es un lenguaje con una sintaxis muy simple pero dotado de un importante y muy potente conjunto de herramientas. Entre las más destacadas están las destinadas a garantizar la correcta estructura de los documentos. XML permite determinar las reglas que debe cumplir un determinado documento escrito en este lenguaje. Estas reglas, junto con los correspondientes procesadores de XML, permiten garantizar que un documento y los datos almacenados en él están correctamente construidos.

4.3.1. XML válido y XML bien formado

Una de las características de un metalenguaje como XML es que no tiene etiquetas predeterminadas. Esto significa que, en lugar de listado de etiquetas con sus significados XML, proporciona una infraestructura sobre la que crear etiquetas y dotarlas de significado. No obstante, hay ciertas reglas generales de diseño que hay que cumplir y que dotan de robustez a los lenguajes creados con XML:

- El documento tiene que estar **bien formado** o, lo que es lo mismo, es **conforme la sintaxis** de XML.
- El documento tiene que ser **válido** o, lo que es lo mismo, **conforme a las reglas** definidas en su DTD *(Document Type Definition)* o XML Schema, aunque estas definiciones de reglas semánticas son opcionales.

Para que un documento esté **bien formado,** tiene que cumplir con las siguientes reglas:

- Debe haber uno y solo un elemento raíz.
- Todos los elementos deben estar cerrados.
- Los elementos tienen que estar anidados correctamente: no se pueden intercalar aperturas y cierres de elementos distintos.
- Todos los valores de los atributos están entrecomillados.
- Los nombres de elementos y atributos han de cumplir con sus respectivas reglas.

Para que un documento sea **válido** tiene que existir una definición que contenga las reglas semánticas que validar. Estas reglas se definen utilizando DTD o XML Schema y se explicarán posteriormente.

Los navegadores y la mayoría de los programas que permiten procesar documentos XML validan que estén bien formados y sean válidos. Si el documento contiene algún error, muestran un mensaje que lo indica y señala su ubicación.

Actividad propuesta 4.2

Documento bien formado

Haz los cambios necesarios en el siguiente documento XML para que esté bien formado:

```
<?xml version="1.0" encoding="UTF-8"?>
<videojuego>
```

© Ediciones Paraninfo

```
    <titulo anyo="2017>Fortnite</titulo>
    <empresa>Epic Games</empresas>
    <motor>Unreal Engine</Motor>
</videojuego>
```

4.3.2. DTD

Se dice que un documento es **válido** si, además de estar **bien formado**, cumple con todas las reglas descritas en un DTD determinado.

Como se ha comentado anteriormente, DTD son las siglas de *Document Type Definition* («definición de tipo de documento», en castellano). Un DTD contiene una serie de reglas que sirven para validar un documento XML. En ningún caso es obligatorio utilizar un DTD para trabajar con documentos XML, pero es recomendable en muchas ocasiones.

Las reglas contenidas en un DTD tienen que ver con la estructura del documento y los elementos y atributos que puede contener. Es un mecanismo relativamente sencillo de usar, aunque tiene algunas limitaciones. Estas limitaciones se pueden superar utilizando XML Schema, otra herramienta de validación de documentos XML.

DTD interna y externa

Existen dos maneras de asociar un DTD a un documento XML: de forma **interna** o **externa.**

En el DTD interno, las reglas se definen dentro del propio documento XML. En el DTD externo, las reglas se definen en un documento externo enlazado con el documento XML.

El DTD interno tiene como ventaja que tanto los datos como las reglas de validación se encuentran en un único documento, lo que en cierta medida puede resultar una alternativa más sencilla y compacta.

El DTD interno se crea incluyendo un elemento de tipo DOCTYPE inmediatamente después de la declaración del documento XML. Este elemento tiene la siguiente sintaxis:

```
<!DOCTYPE nombre-elemento-raíz[
        Elementos-y-sus-relaciones
]>
```

Siendo **nombre-elemento-raíz** el mismo nombre que el del elemento raíz del documento y **Elementos-y-sus-relaciones** el conjunto de elementos y cómo se estructuran y anidan, construyendo así una declaración formal de cómo debe ser un documento XML para ser acorde a un DTD y, por lo tanto, válido.

En el siguiente código se puede observar un DTD interno:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE aviso [
```


© Ediciones Paraninfo

```
  <!ELEMENT aviso (de,para,mensaje)>
  <!ELEMENT de (#PCDATA)>
  <!ELEMENT para (#PCDATA)>
  <!ELEMENT mensaje (#PCDATA)>
]>
<aviso>
    <de>David</de>
    <para>Rosalía</para>
    <mensaje>Mañana nos vemos en el estudio a las 10.</mensaje>
</aviso>
```

Por su parte, el DTD externo facilita la reutilización y permite compartir las reglas de validación para utilizarlas, por ejemplo, en procesos de intercambio de información entre sistemas. De manera general se recomienda, por lo tanto, utilizar DTD externos.

La vinculación de un DTD a un documento XML se realiza añadiendo una referencia al fichero que contiene el DTD en este último. La sintaxis es la siguiente:

```
<!DOCTYPE nombre-elemento-raíz SYSTEM "URI">
```

siendo **nombre-elemento-raíz** el mismo nombre que el del elemento raíz del documento y **URI** la dirección en la que se encuentra el fichero con el DTD.

En el siguiente código se muestra el documento XML en el que se referencia a un DTD externo almacenado en la misma ubicación que el fichero XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE aviso SYSTEM "dtdExterno.dtd">
<aviso>
    <de>David</de>
    <para>Rosalía</para>
    <mensaje>Mañana nos vemos en el estudio a las 10.</mensaje>
</aviso>
```

El documento referenciado, de nombre **dtdExterno.dtd**, contiene los elementos que definen la estructura que debe tener el documento XML:

```
<!ELEMENT aviso (de,para,mensaje)>
<!ELEMENT de (#PCDATA)>
<!ELEMENT para (#PCDATA)>
<!ELEMENT mensaje (#PCDATA)>
```

Como se puede observar, los elementos de definición del DTD son idénticos en la declaración interna y en la externa. Únicamente cambia la ubicación de dichos elementos y, por consecuencia, la forma en la que se hace uso de estos.

© Ediciones Paraninfo

Entidades en los DTD

En un DTD, una entidad es el equivalente a una constante en programación. Cuando se define una entidad, se le asigna un nombre y un valor que sustituirá a las referencias que se hagan a dicho nombre. Una de las categorizaciones divide a las entidades en **internas** y **externas.**

La sintaxis de la declaración de una **entidad interna** es la siguiente:

```
<!ENTITY nombre-entidad "texto de reemplazo">
```

- nombre-entidad: es el alias que se asigna a la entidad.
- «texto de reemplazo»: es cualquier texto (entrecomillado) que sustituirá a las referencias a la entidad.

La sintaxis para referenciar a una entidad interna en el documento XML es la siguiente:

```
&nombre-entidad;
```

En el ejemplo que se muestra a continuación se define una entidad población con texto asociado. Además, se hacen dos referencias en el cuerpo del documento XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE fichas [
  <!ENTITY poblacion "Tejeda de Tiétar, Cáceres">
]>
<fichas>
    <ficha>
        <empresa nombre="Tractores Toledo"></empresa>
        <direccion>&poblacion;</direccion>
    </ficha>
    <ficha>
        <empresa nombre="Congelados Kurylenko"></empresa>
        <direccion>&poblacion;</direccion>
    </ficha>
</fichas>
```

Recuerda

Existen entidades predefinidas que se utilizan para representar caracteres que tienen significado propio en XML y que pueden generar confusión en el análisis.

El resultado al abrir el documento en el navegador muestra la sustitución de las referencias a la entidad por el texto asociado:

© Ediciones Paraninfo

```
–<fichas>
  –<ficha>
      <empresa nombre="Tractores Toledo"/>
      <direccion>Tejeda de Tiétar, Cáceres</direccion>
    </ficha>
  –<ficha>
      <empresa nombre="Congelados Kurylenko"/>
      <direccion>Tejeda de Tiétar, Cáceres</direccion>
    </ficha>
  </fichas>
```

Figura 4.6. *Las entidades permiten la reutilización de los datos.*

Por su parte, las **entidades externas** se definen fuera del DTD. Se dividen en **públicas** y **privadas.**

Las **entidades externas públicas** tienen la siguiente sintaxis:

```
<!ENTITY nombre-entidad PUBLIC "id-publico" "URI/URL">
```

- **nombre-entidad:** es el alias que se asigna a la entidad.
- **PUBLIC:** palabra clave que indica que la entidad es pública.
- **id-publico:** es el identificador público.
- **URI/URL:** es el identificador o la ubicación del recurso.

Las **entidades externas privadas** o de sistema, por su parte, tienen la siguiente sintaxis:

```
<!ENTITY nombre-entidad SYSTEM "URI/URL">
```

- **nombre-entidad:** es el alias que se asigna a la entidad.
- **SYSTEM:** palabra clave que indica que la entidad es privada.
- **URI/URL:** es el identificador o la ubicación del recurso.

En el siguiente ejemplo se muestra la aplicación de entidades internas y externas:

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE fichas [
  <!ENTITY poblacion "Tejeda de Tiétar">
  <!ENTITY provincia SYSTEM "provincia.txt">
  <!ENTITY detalle SYSTEM "ficha.xml">
]>
<grupo>
  &detalle;
  <fichas>
    <ficha>
      <empresa nombre="Tractores Toledo"></empresa>
     <poblacion>&poblacion;</poblacion>
```

© Ediciones Paraninfo

```
      <provincia>&provincia;</provincia>
    </ficha>
    <ficha>
      <empresa nombre="Congelados Kurylenko"></empresa>
      <poblacion>&poblacion;</poblacion>
      <provincia>&provincia;</provincia>
    </ficha>
  </fichas>
</grupo>
```

Las entidades externas tienen referencias a sendos ficheros escritos en texto sin formato (provincia.txt) y en formato XML (ficha.xml). Son los siguientes:

provincia.txt:

```
Cáceres
```

ficha.xml:

```
<distribuidores>
    <codigo>10834</codigo>
    <descripcion>Distribuidores oficiales</descripcion>
</distribuidores>
```

El resultado al abrir el fichero en un navegador es:

```
▼<grupo>
  ▼<fichas>
    ▼<ficha>
       <empresa nombre="Tractores Toledo"/>
       <poblacion>Tejeda de Tiétar</poblacion>
       <provincia/>
     </ficha>
    ▼<ficha>
       <empresa nombre="Congelados Kurylenko"/>
       <poblacion>Tejeda de Tiétar</poblacion>
       <provincia/>
     </ficha>
   </fichas>
 </grupo>
```

Figura 4.7. *Las entidades externas admiten múltiples formatos.*

Como se puede observar, la inclusión de las entidades externas no se ha realizado por restricciones incorporadas a los navegadores por razones de seguridad.

En cambio, al abrir el fichero desde el programa XML Notepad, se puede observar que las entidades externas se han incluido en el fichero.

© Ediciones Paraninfo

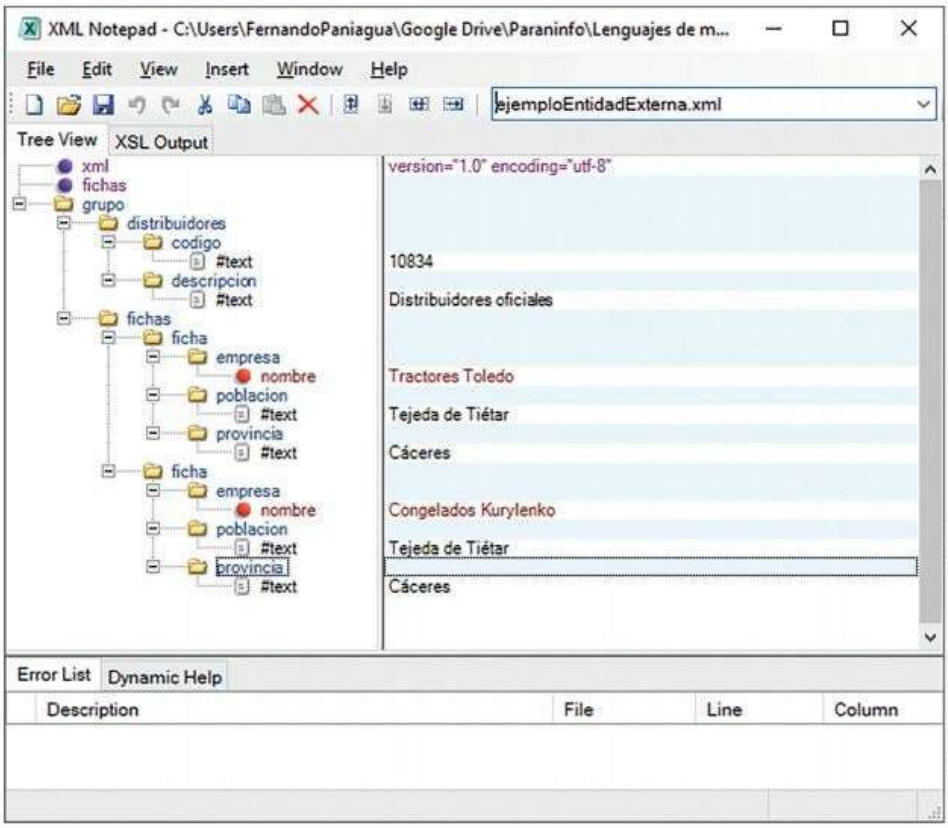


Figura 4.8. *Solo algunos programas procesan las entidades externas.*

Nota técnica

Existe un tipo de ataque informático llamado «ataque de entidad externa XML», que se realiza sobre la aplicación que analiza un fichero XML que contiene una referencia a una entidad externa.

Anotaciones

Las anotaciones (**notation**) se utilizan para identificar el formato de entidades que no son XML y que, por lo tanto, no se van a procesar, o como valor válido para un atributo. Pueden ser públicas o privadas.

La sintaxis general es cualquiera de las siguientes:

<!NOTATION nombre SYSTEM "URI">

<!NOTATION nombre PUBLIC "id-público" >

<!NOTATION nombre PUBLIC "id-público" "URI">

Ejemplo:

```
<!NOTATION gif PUBLIC "GIF">
```

© Ediciones Paraninfo

Elementos

Al incluir un elemento en un DTD, se está indicando en qué condiciones debe encontrarse dicho elemento en el documento XML que se está validando.

Por ejemplo, si en un DTD se incluye la siguiente declaración:

```
<!ELEMENT de (#PCDATA)>
```

Se está indicando que el documento XML debe tener un elemento <de> que contiene una cadena de caracteres.

La declaración de elementos en un DTD se realiza según la siguiente sintaxis:

```
<!ELEMENT nombre-elemento contenido>
```

siendo **nombre-elemento** el nombre del elemento referenciado, y **contenido** la especificación del tipo contenido válido para dicho elemento.

Los posibles tipos de **contenido** son los que se detallan en la Tabla 4.3.

Tabla 4.3. Posibles tipos de contenido de los elementos

Tipo de contenido	Descripción
EMPTY	Indica que el elemento referenciado debe estar vacío.
ANY	Señala que el elemento referenciado puede contener cualquier contenido.
(#PCDATA)	Informa de que el elemento referenciado puede tener datos de tipo carácter *(Parsed Character Data)*.
(*nombreElemento*)	Indica que el elemento referenciado puede contener al elemento indicado.
(*nombreElemento1, nombreElemento2, …*)	Señala que el elemento referenciado puede contener una sucesión de elementos indicados como una lista separada por comas.

En las relaciones entre elementos se pueden especificar cardinalidades para determinar cuántos elementos pueden estar contenidos dentro del elemento referenciado. Las opciones son las que se recogen en la Tabla 4.4.

Tabla 4.4. Cardinalidades en las relaciones

Notación	Descripción	Ejemplo
(*nombreElemento*)	Una única ocurrencia del elemento.	<!ELEMENT aviso (de)>
(*nombreElemento?*)	Cero o una única ocurrencia del elemento.	<!ELEMENT aviso (de?)>

© Ediciones Paraninfo

Notación	Descripción	Ejemplo
(*nombreElemento+*)	Una o más ocurrencias del elemento.	<!ELEMENT aviso (mensaje+)>
(*nombreElemento**)	Cero o más ocurrencias del elemento.	<!ELEMENT aviso (mensaje*)>
(*nombreElemento1, nombreElemento2,…*)	Debe contener todos los elementos de la lista.	<!ELEMENT aviso (de, para, mensaje)>
(nombreElemento1 | nombreElemento2)	Debe contener uno u otro elemento.	<!ELEMENTO aviso (de | para, mensaje)>

Estas opciones para la declaración de relaciones entre elementos se pueden combinar y agrupar con paréntesis para asignar las cardinalidades. Por ejemplo, se puede declarar un elemento de la siguiente manera:

```
<!ELEMENT aviso (#PCTADA | mensaje | nota | correo | llamada)*>
```

El significado para el ejemplo anterior es el siguiente:

Un aviso puede tener cero o más ocurrencias del tipo texto libre, elemento mensaje, elemento nota, elemento correo o elemento llamada.

A continuación, se expone un ejemplo de un DTD interno en un documento XML que es válido respecto a dicho DTD:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE cv [
  <!ELEMENT cv (nombre, direccion, telefono, fax?, email+, idiomas)>
  <!ELEMENT nombre (#PCDATA)>
  <!ELEMENT direccion (#PCDATA)>
  <!ELEMENT telefono (#PCDATA)>
  <!ELEMENT fax (#PCDATA)>
  <!ELEMENT email (#PCDATA)>
  <!ELEMENT idiomas (idioma*)>
  <!ELEMENT idioma (#PCDATA)>
]>
<cv>
    <nombre>Desmond Miles</nombre>
    <direccion>Black Hills, Dakota del Sur, Estados Unidos</direccion>
    <telefono>+1-555-73-66-58</telefono>
    <email>desmond.miles@servidordeprueba.org</email>
    <email>desmond.miles@otroservidor.fr</email>
    <idiomas>
        <idioma>Francés</idioma>
        <idioma>Inglés</idioma>
        <idioma>Italiano</idioma>
    </idiomas>
</cv>
```

© Ediciones Paraninfo

La explicación de las condiciones de validación del DTD anterior se muestran en la Tabla 4.5.

Tabla 4.5. Significado de las reglas del DTD de ejemplo

Definición	Descripción
<!DOCTYPE cv	Indica que **cv** es el elemento raíz del documento.
<!ELEMENT cv (nombre, direccion, telefono, fax?, email+, idiomas)>	Señala que el elemento **cv** tiene que contener los siguientes elementos: un **nombre,** una **dirección,** un **teléfono,** opcionalmente un **fax,** uno o más **email** y un **idiomas.**
<!ELEMENT nombre (#PCDATA)>	Informa de que el elemento **nombre** puede contener texto.
<!ELEMENT direccion (#PCDATA)>	Indica que el elemento **direccion** puede contener texto.
<!ELEMENT telefono (#PCDATA)>	Señala que el elemento **telefono** puede contener texto.
<!ELEMENT fax (#PCDATA)>	Indica que el elemento **fax** puede contener texto.
<!ELEMENT email (#PCDATA)>	Informa de que el elemento **email** puede contener texto.
<!ELEMENT idiomas (idioma*)>	Señala que el elemento **idiomas** opcionalmente (podría no tener ninguno) puede tener varios elementos **idioma.**
<!ELEMENT idioma (#PCDATA)>	Indica que el elemento **idioma** puede contener texto.

Respecto a los atributos, los DTD también permiten incluir reglas y condiciones que determinen si un documento XML es válido. Se declaran a nivel de elemento. La sintaxis es la siguiente:

```
<!ATTLIST nombre-elemento nombre-atributo tipo-atributo valor-atributo>
```

Ejemplo:

```
<!ATTLIST direccion calle CDATA "Domicilio desconocido">
```

La lista de los tipos de atributos más utilizados se recoge en la Tabla 4.6.

Tabla 4.6. Tipos de atributos

Tipo	Descripción
CDATA	Cadena de caracteres.
(*valor1* | *valor2* | …)	Una lista de posibles valores.
ID	Un identificador único.
IDREF	Una referencia a un identificador único de otro elemento.

© Ediciones Paraninfo

Tipo	Descripción
IDREFS	Una lista de referencias separadas por espacios a identificadores de otros elementos.
NMTOKEN	Un nombre XML válido.
NMTOKENS	Una lista de nombres XML válidos separados por espacios.
ENTITY	Una referencia a una entidad.
ENTITIES	Una referencia a un conjunto de entidades.
NOTATION	Un nombre de una notación.
xml:lang	Indica el idioma del contenido.
xml:space	Señala si se han de respetar los espacios, las tabulaciones y los retornos de carro múltiples (valor «preserve») o eliminarlos (valor «default»).

La lista de valores posibles se puede consultar en la Tabla 4.7.

Tabla 4.7. Valores posibles para los atributos

Valor	Descripción
valor	El valor por defecto del atributo.
#REQUIRED	Indica que el atributo es obligatorio.
#IMPLIED	Señala que el atributo es opcional.
#FIXED *valor*	El valor del atributo.

Por ejemplo, la siguiente declaración de atributos indica que el elemento **idioma** tiene un atributo **nivel** que puede tomar los valores **bajo, medio** y **alto,** asignándosele por defecto el valor **medio.**

```
<!ATTLIST idioma nivel (bajo | medio | alto) "medio">
```

Aplicado al siguiente fragmento de XML:

```
<idiomas>
    <idioma nivel="bajo">Francés</idioma>
    <idioma>Inglés</idioma>
    <idioma>Italiano</idioma>
</idiomas>
```

genera la salida en el navegador que se recoge en la Figura 4.9.

© Ediciones Paraninfo

```
−<cv>
    <nombre>Desmond Miles</nombre>
    <direccion>Black Hills, Dakota del Sur, Estados Unidos</direccion>
    <telefono>+1-555-73-66-58</telefono>
    <email>desmond.miles@servidordeprueba.org</email>
    <email>desmond.miles@otroservicor.fr</email>
  −<idiomas>
      <idioma nivel="bajo">Francés</idioma>
      <idioma nivel="medio">Inglés</idioma>
      <idioma nivel="medio">Italiano</idioma>
    </idiomas>
  </cv>
```

Figura 4.9. *Se puede definir un valor por defecto para los atributos.*

Como se puede observar en la Figura 4.9, aquellos elementos **idioma** a los que no se les ha asignado atributos **nivel** se les asigna de manera automática, al estar indicado un valor por defecto en la especificación del atributo en el DTD.

Si se modifica la declaración del atributo en el DTD sustituyendo el valor por defecto por #REQUIRED, cuando se realice una validación del documento se mostrarán tantos errores como elementos **idioma** carezcan del atributo **nivel** (Figura 4.10).

```
<!ATTLIST idioma nivel (bajo | medio | alto) #REQUIRED>
```

```
19 <idiomas>
20     <idioma nivel="bajo">Francés</idioma>
21     <idioma>Inglés</idioma>   Attribute "nivel" is required and must be specified for element type "idioma".
22     <idioma>Italiano</idioma>
23 </idiomas>
```

Figura 4.10. *Si el atributo obligatorio no existe, el procesador muestra un error.*

En paralelo, además de validar que los elementos y atributos existen y están en el orden preciso, se valida que no hay elementos o atributos no declarados en el DTD.

En la Figura 4.11 se muestra un documento XML con el DTD interno. El editor utilizado, al realizar la validación, marca errores en las líneas 9, 11, 13, 14 y 15.

Los errores de la línea 9 y 11 tienen el mismo origen: el elemento **categoria** no está declarado como elemento hijo del elemento **cv.** Esto provoca por un lado que el elemento **cv** no esté bien construido, provocando un error. Por otra parte, el elemento **categoria** no aparece como declarado en ningún lugar, por lo que provoca otro error.

Los errores de las líneas 13, 14 y 15 tienen el mismo origen: el atributo **código** no está definido como atributo del elemento **idioma.**

© Ediciones Paraninfo

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <!DOCTYPE cv [
3   <!ELEMENT cv (nombre, idiomas)>
4   <!ELEMENT nombre (#PCDATA)>
5   <!ELEMENT idiomas (idioma*)>
6   <!ELEMENT idioma (#PCDATA)>
7   <!ATTLIST idioma nivel (bajo | medio | alto) "medio">
8 ]>
9 <cv>
10     <nombre>Desmond Miles</nombre>
11     <categoria>Senior</categoria>
12     <idiomas>
13         <idioma codigo="fr" nivel="bajo">Francés</idioma>
14         <idioma codigo="en">Inglés</idioma>
15         <idioma codigo="it">Italiano</idioma>
16     </idiomas>
17 </cv>
```

Figura 4.11. *En caso de no declarar un atributo en DTD, su existencia provocará errores.*

En el siguiente ejemplo se muestra un uso de los tipos de atributo **ID** e **IDREFS.** Se declara un atributo **identificador** en el elemento **lengua** y se le asigna el tipo **ID,** convirtiendo al atributo en un identificador de entidad (los valores no se pueden repetir). Por otra parte, se declara el atributo **codigoPais** del elemento **idioma** como una referencia a un identificador de otra entidad (sin especificar cuál) mediante el tipo **IDREF.** Como resultado de esta declaración, para que el documento XML sea válido, los valores de los atributos **codigoPais** deben coincidir con los valores de atributos que sean identificadores de otra entidad.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE cv [
  <!ELEMENT cv (nombre, idiomas, lenguas)>
  <!ELEMENT nombre (#PCDATA)>
  <!ELEMENT idiomas (idioma*)>
  <!ELEMENT idioma (#PCDATA)>
  <!ELEMENT lenguas (lengua*)>
  <!ELEMENT lengua (#PCDATA)>
  <!ATTLIST lengua identificador ID #REQUIRED>
  <!ATTLIST idioma nivel (bajo | medio | alto) "medio">
  <!ATTLIST idioma codigoPais IDREF #REQUIRED>
]>
<cv>
    <nombre>Desmond Miles</nombre>
    <idiomas>
        <idioma codigoPais="fr" nivel="bajo">Francés</idioma>
        <idioma codigoPais="en">Inglés</idioma>
        <idioma codigoPais="it">Italiano</idioma>
    </idiomas>
    <lenguas>
        <lengua identificador="fr">Francés</lengua>
        <lengua identificador="en">Inglés</lengua>
        <lengua identificador="it">Italiano</lengua>
    </lenguas>
</cv>
```

© Ediciones Paraninfo

Actividad resuelta 4.2

Validación de receta de cocina con DTD

Crear el DTD que valide un documento XML que almacene recetas de cocina. La estructura es la siguiente:

- El elemento principal es la receta.
- Una receta está formada por el nombre (un texto libre), un listado con uno o más ingredientes y un texto con las instrucciones de elaboración (un texto libre).
- Un ingrediente contiene texto libre.
- Las recetas tienen un atributo que indica el número de raciones.
- Las recetas tienen un atributo que indica la dificultad, aceptando los valores baja, media (por defecto) y alta.
- Los ingredientes tienen un atributo opcional con la cantidad del ingrediente.
- Los ingredientes tienen un atributo opcional con la unidad en la que están expresadas las cantidades.

Solución

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE receta [
  <!ELEMENT receta (nombre,ingredientes,elaboracion)>
  <!ELEMENT nombre (#PCDATA)>
  <!ELEMENT ingredientes (ingrediente+)>
  <!ELEMENT elaboracion (#PCDATA)>
  <!ELEMENT ingrediente (#PCDATA)>

  <!ATTLIST receta raciones CDATA #REQUIRED>
  <!ATTLIST receta dificultad (baja | media | alta) "media">
  <!ATTLIST ingrediente cantidad CDATA #IMPLIED>
  <!ATTLIST ingrediente unidad CDATA #IMPLIED>
]>
<receta raciones="6" dificultad="media">
    <nombre>Tortilla de patatas</nombre>
    <ingredientes>
        <ingrediente cantidad="1" unidad="kilogramo">Patata</ingrediente>
        <ingrediente cantidad="8" unidad="unidades">Huevo</ingrediente>
        <ingrediente cantidad="1" unidad="unidades">Cebolla</ingrediente>
        <ingrediente>Aceite de Oliva</ingrediente>
        <ingrediente>Sal</ingrediente>
    </ingredientes>
    <elaboracion>
    Se pelan las patatas...
    </elaboracion>
</receta>
```

Actividad propuesta 4.3

Validación de una especie animal con DTD

Crea el DTD que valide un XML que representa una especie animal y que tiene la siguiente estructura:

© Ediciones Paraninfo

- El elemento principal contiene un atributo **estado** que puede tomar los valores **extinto, amenazado** y **sin-peligro.**
- El elemento principal contiene un nombre (texto), una familia (texto) y un listado de áreas de distribución.
- El nombre contiene un atributo con el nombre científico.
- El listado de áreas de distribución está formado por, al menos, una zona (texto).

XML de ejemplo:

```
<?xml version="1.0" encoding="UTF-8"?>
<especie estado="amenazado">
    <nombre nombre-cientifico="Monachus monachus">Foca monje</nombre>
    <familia>Phocidae</familia>
    <distribucion>
        <zona>Costa Atlántica</zona>
        <zona>Costa Mediterránea</zona>
    </distribucion>
</especie>
```

4.3.3. XML Schema

XML Schema («esquema XML», en castellano) es el nombre con el que se conoce habitualmente al lenguaje utilizado para describir la estructura, relaciones y restricciones de los documentos XML. El nombre técnico es XML Schema Definition (XSD), pero su uso es menos frecuente salvo en entornos más académicos o especializados. En cualquier caso, las referencias a XML Schema y a XSD suelen ser al mismo concepto.

XML Schema permite conseguir un mayor nivel de precisión en el establecimiento de las reglas de validación, consiguiendo un nivel de detalle que no es posible alcanzar con los DTD.

La primera versión publicada como recomendación por el W3C data del año 2001 y surge para cubrir las carencias de los DTD, incorporando, por ejemplo, diversos tipos de datos simples y complejos o cardinalidades más precisas en las relaciones entre elementos. La definición de XML Schema se divide en tres partes, cada una de las cuales describe una parte de la especificación: XML Schema Part 0: Primer; XML Schema Part 1: Structures, y XML Schema Part 2: Datatypes.

Las aportaciones más relevantes que XML Schema proporciona son las siguientes:

- Permite determinar qué elementos y atributos admite un XML.
- Permite determinar qué tipos de datos pueden utilizar los elementos y atributos, habilitando la creación de nuevos tipos.
- Permite asignar valores por defecto a elementos y atributos.
- Permite determinar cardinalidades precisas.
- Permite determinar las relaciones entre los elementos del documento XML: qué hijos tiene cada elemento, en qué número y en qué orden.

© Ediciones Paraninfo

Los documentos XML Schema están escritos en XML y se suelen almacenar con la extensión .xsd.

Nota técnica

Con el programa **Liquid Studio** es muy sencillo validar un documento XML utilizando XML Schema. Abriendo el fichero que se quiere validar y pulsando la tecla F7, el programa realiza la validación y notifica los errores, siempre y cuando el fichero XSD esté correctamente referenciado.

Estructura básica

A continuación, se muestra un ejemplo en el que se pueden observar las dos partes del uso de XML Schema: el documento XML que se quiere validar (en este caso el resultado de la validación es positivo) y el documento XSD (también en formato XML) que contiene la estructura y restricciones.

En el documento XML se debe incluir una referencia al documento XSD que contiene las reglas de validación:

```
<?xml version="1.0" encoding="UTF-8"?>
<mensaje
    xmlns:xs="http://www.w3.org/2001/XMLSchema-instance"
    xs:noNamespaceSchemaLocation="mensaje.xsd">
    <de>Anna</de>
    <para>Rocío</para>
    <mensaje>Ya estoy en la biblioteca.</mensaje>
</mensaje>
```

En el documento XSD (XML Schema) se define la estructura y se indican los elementos que debe contener el documento XML y de qué tipo son cada uno:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
    <xs:element name="mensaje">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="de" type="xs:string"/>
                <xs:element name="para" type="xs:string"/>
                <xs:element name="mensaje" type="xs:string"/>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
</xs:schema>
```

La estructura básica se compone de las declaraciones del documento XML, del elemento raíz y del espacio de nombres, y de los elementos y sus relaciones:

© Ediciones Paraninfo

- Declaración del documento XML.

```
<?xml version="1.0" encoding="UTF-8"?>
```

- Declaración del elemento raíz y del espacio de nombres. El elemento raíz es **<schema>**. El espacio de nombres se asigna mediante el atributo **xmlns**, y habitualmente se le asocia el prefijo **xs**, aunque también es frecuente encontrar el prefijo **xsd**.

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
```

- Declaración de los elementos. En esta sección se indican los elementos que componen la estructura del XML, su jerarquía, relación, atributos y demás restricciones. Los elementos se pueden clasificar de varias maneras:
  - Elementos **locales** y elementos **globales**.
  - Elementos **simples** y elementos **complejos**.
- Declaración de los atributos. Asociados a los elementos, se determina qué atributos y qué valores son admitidos.

Elementos locales y elementos globales

Una primera clasificación de los elementos los categoriza como **locales y globales**.

Los elementos locales son hijos de elementos que no son el elemento raíz y solo se pueden utilizar una vez.

Los elementos globales son hijos del elemento raíz y pueden ser reutilizados.

Elementos simples

Los elementos de tipo **simple** solo pueden contener texto, aunque esta restricción debe explicarse. Los tipos de datos que se admiten pueden ser cadenas de caracteres, fechas, valores lógicos, etc. La restricción hace referencia a que el elemento tendrá un valor textual que representará cualquiera de estos tipos. No podrán contener a otros elementos ni tener atributos.

La sintaxis básica de un elemento es la siguiente:

```
<xs:element name=""></xs:element>
```

Siendo **element** la etiqueta que determina que es un elemento y el atributo **name** el que determina el nombre de este.

El siguiente código muestra la declaración de dos elementos simples:

```
<xs:element name="email" type="xs:string"/>
<xs:element name="edad" type="xs:integer"/>
```

© Ediciones Paraninfo

Además de **name**, algunos de los atributos que se pueden asignar para determinar cómo ha de ser un elemento son los siguientes:

- **type.** Tipo de dato.
- **default.** Permite asignar un valor por defecto.
- **fixed.** Determina el valor del atributo en caso de que este exista.
- **minOccurs.** Número mínimo de ocurrencias del elemento. El valor por defecto es 1. Si se quiere que el número mínimo de ocurrencias sea ilimitado, el valor que debe tomar es «unbounded».
- **maxOccurs.** Número máximo de ocurrencias del elemento. El valor por defecto es 1. Si se quiere que el número máximo sea ilimitado, el valor que debe tomar es «unbounded».

Elementos complejos

Los elementos de tipo **complejo** pueden actuar de contenedor. Se declaran utilizando la etiqueta **complexType** y pueden:

- Contener a otros elementos, pudiendo contener opcionalmente atributos.
- Estar vacíos, pudiendo contener opcionalmente atributos.
- Tener contenido mixto: otros elementos y texto, pudiendo contener opcionalmente atributos.

La sintaxis básica es la siguiente:

```
<xs:element name="" type="">
    <xs:complexType/>
    </xs:complexType>
</xs:element>
```

A continuación, se muestran un XML y el XSD que lo valida y que hace uso de varios tipos de elementos complejos:

```
<?xml version="1.0" encoding="UTF-8"?>
<factura
    xmlns:xs="http://www.w3.org/2001/XMLSchema-instance"
    xs:noNamespaceSchemaLocation="factura.xsd">
    <numero>1</numero>
    <cliente>
      <nombre>Ediciones Paraninfo S.A.</nombre>
      <CIF>A81461477</CIF>
    </cliente>
    <concepto>
      <descripcion>Imprenta</descripcion>
     <importe>1000</importe>
```


© Ediciones Paraninfo

```
    </concepto>
    <concepto>
      <descripcion>Maquetación</descripcion>
      <importe>500.50</importe>
    </concepto>
    <impuesto nombre="IVA">PENDIENTE</impuesto>
    <totalFactura euros="1500.50"/>
</factura>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
    <xs:element name="factura">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="numero" type="xs:string"/>
                <xs:element name="cliente" type="cliente"/>
                <xs:element name="concepto" type="concepto" minOccurs ="0"
maxOccurs="unbounded" />
                <xs:element name="impuesto">
                    <xs:complexType mixed="true">
                        <xs:attribute name="nombre" type="xs:string"/>
                    </xs:complexType>
                </xs:element>
                <xs:element name="totalFactura" type="totalFactura"/>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
    <xs:complexType name="cliente">
        <xs:sequence>
            <xs:element name="nombre"/>
            <xs:element name="CIF"/>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="concepto">
        <xs:sequence>
            <xs:element name="descripcion"/>
            <xs:element name="importe"/>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="totalFactura">
        <xs:attribute name="euros" type="xs:decimal"/>
    </xs:complexType>
</xs:schema>
```

En este bloque se declara un elemento complejo global contenedor de otros elementos. El elemento **cliente** contiene dos elementos **nombre** y **CIF:**

© Ediciones Paraninfo

```
<xs:complexType name="cliente">
        <xs:sequence>
          <xs:element name="nombre"/>
          <xs:element name="CIF"/>
        </xs:sequence>
</xs:complexType>
```

A continuación, se muestra el código XML válido para el bloque XML Schema anterior:

```
<cliente>
        <nombre>Ediciones Paraninfo S.A.</nombre>
        <CIF>A81461477</CIF>
</cliente>
```

En este bloque se declara un elemento global complejo vacío. El elemento **totalFactura** únicamente tiene un atributo **euros** de tipo **xs:decimal:**

```
<xs:complexType name="totalFactura">
        <xs:attribute name="euros" type="xs:decimal"/>
</xs:complexType>
```

A continuación, se muestra el código XML válido para el bloque XML Schema anterior:

```
<totalFactura euros="1500.50"/>
```

En este bloque se declara un elemento local complejo mixto. El elemento **impuesto** contiene un atributo **nombre** de tipo **xs:string** y, opcionalmente, un bloque de texto.

```
<xs:element name="impuesto">
    <xs:complexType mixed="true">
        <xs:attribute name="nombre" type="xs:string"/>
    </xs:complexType>
</xs:element>
```

A continuación, se muestra el código XML válido para el bloque XML Schema anterior:

```
<impuesto nombre="IVA">PENDIENTE</impuesto>
```

Subelementos

Los elementos pueden contener subelementos. Existen tres tipos:

- **xs:sequence.** Indica una secuencia de elementos obligatorios. Deben aparecer todos y en el mismo orden.
- **xs:choise.** Señala una secuencia de elementos alternativos. Solo debe aparecer uno.

© Ediciones Paraninfo

- **xs:all.** Indica una secuencia de elementos opcionales. No es obligatorio que aparezcan todos ni en el mismo orden.

En el siguiente código XSD se indica que en el XML que valida debe existir un elemento **<direccion>** que contenga o el elemento **<particular>** o el elemento **<profesional>**.

```
<xs:element name="direccion">
    <xs:complexType>
        <xs:choice>
            <xs:element name="particular" type="xs:string"/>
            <xs:element name="profesional" type="xs:string"/>
        </xs:choice>
    </xs:complexType>
</xs:element>
```

Este fragmento de documento XML sería válido según la regla anterior:

```
<direccion>
    <particular>Calle San Miguel, 52</particular>
</direccion>
```

En cambio, el siguiente fragmento provocará un error, ya que el subelemento **choice** solo acepta uno de los elementos de la secuencia:

```
<direccion>
    <particular>Calle San Miguel, 52</particular>
    <profesional>Calle Rincón del Prado, 28</profesional>
</direccion>
```

Atributos

Los atributos tienen la siguiente sintaxis básica:

```
<xs:attribute name="" type=""/>
```

Las características principales de los atributos son las siguientes:

- Deben aparecer inmediatamente después de la declaración del elemento al que hacen referencia.
- El atributo **name** indica el nombre del atributo.
- El atributo **type** indica el tipo del atributo. Deben ser de tipo simple. El tipo por defecto es **anySimpleType**, un tipo de dato genérico que admite cualquier tipo simple.
- El atributo **use** hace referencia a su obligatoriedad, admitiendo los valores **required** (obligatorio), **optional** (opcional, valor por defecto) y **prohibited** (el atributo no se puede utilizar en el elemento).

© Ediciones Paraninfo

- El atributo **default** permite asignar un valor por defecto.
- El atributo **fixed** determina el valor del atributo en caso de que este exista.
- No tienen orden de aplicación.
- No tienen cardinalidad.
- No pueden tener hijos.
- Los tipos de valores de los atributos se pueden restringir con las **restricciones.**

En el siguiente fragmento de documento XSD se muestra la declaración de dos atributos (**nombre** de tipo **string** y **urgencia** de tipo **integer**) pertenecientes al elemento **destinatario.**

```
<xs:element name="destinatario" maxOccurs="unbounded">
    <xs:complexType>
        <xs:attribute name="nombre" type="xs:string"/>
        <xs:attribute name="urgencia" type="xs:integer"/>
    </xs:complexType>
</xs:element>
```

Restricciones

En XML Schema se pueden restringir los valores que pueden tomar los elementos y los atributos. A estas restricciones se les denomina **facetas.** Algunas de las facetas disponibles se recogen en la Tabla 4.8.

Tabla 4.8. Facetas de XML Schema

Faceta	Descripción
xs:length	Determina una longitud fija.
xs:minLength	Establece una longitud mínima.
xs:maxLength	Fija una longitud máxima.
xs:totalDigits	Determina el máximo número de dígitos que puede tener un número.
xs:fractionDigits	Establece el máximo número de decimales que puede tener un número.
xs:minExclusive	Determina que el valor debe ser mayor que el valor indicado.
xs:maxExclusive	Establece que el valor debe ser menor que el valor indicado.
xs:minInclusive	Fija que el valor debe ser mayor o igual que el valor indicado.
xs:maxInclusive	Determina que el valor debe ser menor o igual que el valor indicado.
xs:enumeration	Establece una lista de valores posibles.
xs:whiteSpace	Determina cómo tratar los espacios en blanco, las tabulaciones y los saltos de línea.
xs:pattern	Fija un patrón de caracteres permitidos.

© Ediciones Paraninfo

Ejemplo de restricción de mínimo y máximo:

```
<xs:element name="diaSemana">
    <xs:simpleType>
        <xs:restriction base="xs:integer">
            <xs:minInclusive value="1"/>
            <xs:maxInclusive value="7"/>
        </xs:restriction>
    </xs:simpleType>
</xs:element>
```

Un posible elemento correcto en el XML sería el siguiente:

```
<diaSemana>7</diaSemana>
```

Ejemplo de restricción de enumeración:

```
<xs:element name="diaSemana">
    <xs:simpleType>
        <xs:restriction base="xs:string">
            <xs:enumeration value="lunes"/>
            <xs:enumeration value="martes"/>
            <xs:enumeration value="miércoles"/>
            <xs:enumeration value="jueves"/>
            <xs:enumeration value="viernes"/>
            <xs:enumeration value="sábado"/>
            <xs:enumeration value="domingo"/>
        </xs:restriction>
    </xs:simpleType>
</xs:element>
```

Un posible elemento correcto en el XML sería el siguiente:

```
<diaSemana>sábado</diaSemana>
```

Ejemplo de restricción basada en un patrón:

```
<xs:element name="clave">
    <xs:simpleType>
        <xs:restriction base="xs:string">
            <xs:pattern value="[a-zA-Z]{4}[0-9]{2}"/>
        </xs:restriction>
    </xs:simpleType>
</xs:element>
```

La expresión indicada en el **value** significa que se requieren cuatro letras, ya sean estas minúsculas o mayúsculas seguidas de dos dígitos.

© Ediciones Paraninfo

Un posible elemento correcto en el XML sería el siguiente:

```
<clave>PanI91</clave>
```

Tipos de datos

Una de las diferencias más evidentes de XML Schema frente a DTD consiste en la posibilidad de indicar el tipo de dato preciso que contiene un elemento.

Los tipos de datos más comunes se muestran en la Tabla 4.9.

Tabla 4.9. Tipos de datos en XML Schema

Tipo (utilizando xs como prefijo)	Descripción
xs:string	Cadena de caracteres.
xs:integer	Números enteros.
xs:decimal	Números decimales, utilizando el punto como separador decimal.
xs:boolean	Tipo de dato lógico. Admite los valores «true» y «false».
xs:date	Fechas en formato «AAAA-MM-DD» donde **AAAA** indica el año, **MM** el mes y **DD** el día del mes.
xs:time	Horas en formato «hh:mm:ss» donde **hh** indica la hora, **mm** el minuto y **ss** el segundo.
xs:duration	Un periodo de tiempo en formato «PnYnMnDTnHnMnS», donde **P** indica un periodo; **nY,** número de años: **nM,** número de meses; **nD,** número de días; **T,** el comienzo de la sección referente al tiempo; **nH,** número de horas; **nM,** número de minutos, y **nS,** número de segundos.

La relación de tipos de datos es mucho más extensa. En la Figura 4.12, obtenida de la web del W3C, se muestran dichos tipos con su jerarquía y relaciones.

XML Schema permite crear tipos derivados a partir de tipos existentes. Existen dos mecanismos para la creación o derivación de los tipos de datos: por **restricción** y por **extensión.**

La creación por restricción tiene la siguiente sintaxis:

```
<xs:restriction base=""></xs:restriction>
```

La creación por extensión solo permite crear tipos de datos complejos y tiene la siguiente sintaxis:

```
<xs:extension base=""></xs:extension>
```

En ambos casos el valor del atributo **base** es el tipo del que se está derivando.

© Ediciones Paraninfo

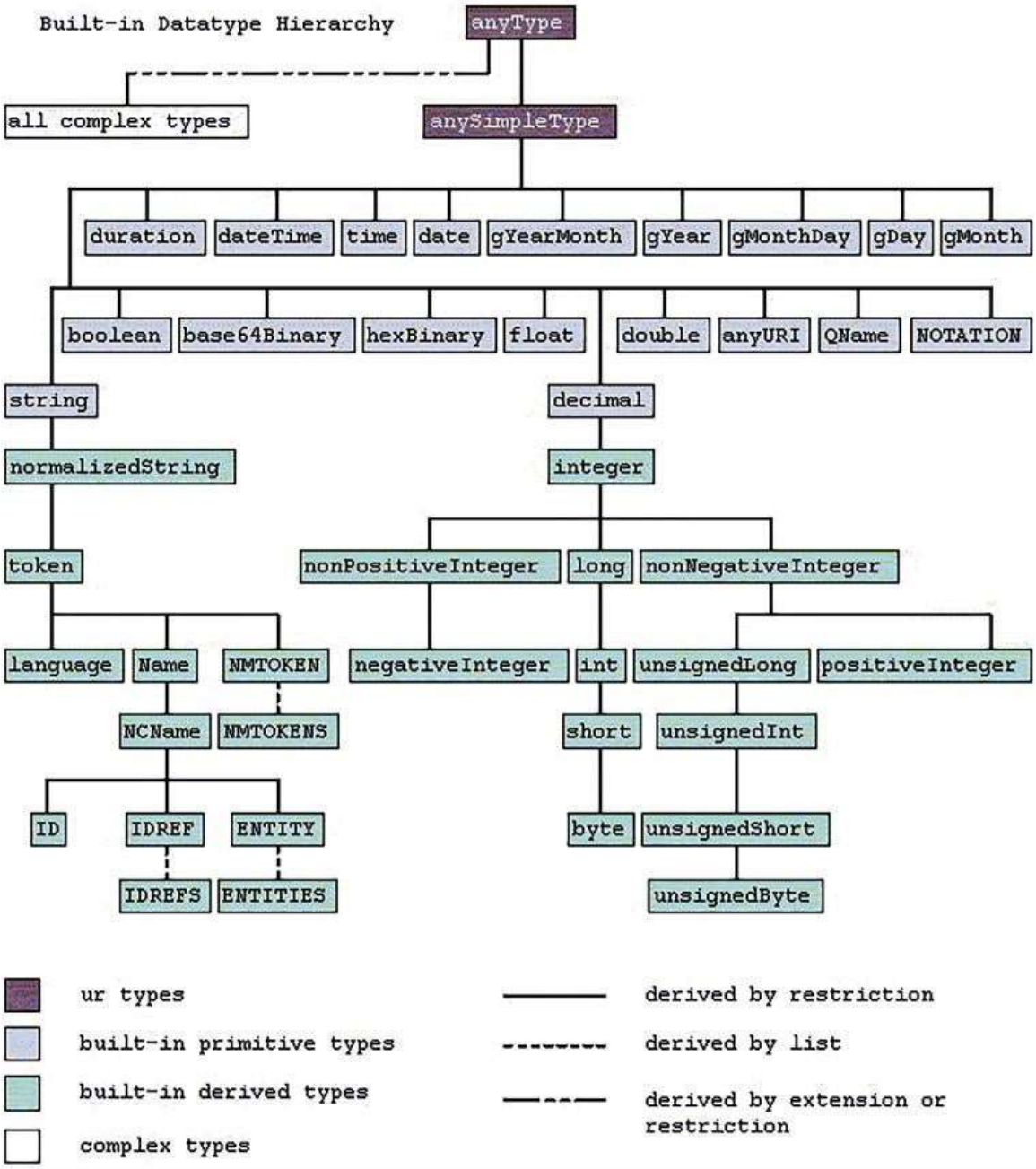


Figura 4.12. *Una de las características de XML Schema es la gran cantidad de tipos de datos.*

El atributo **final** determina la capacidad de extensión que tiene un tipo. Los valores posibles son:

- **extension:** impide elementos derivados por extensión.
- **restriction:** impide elementos derivados por restricción.
- **#all:** impide elementos derivados.

Nota técnica

La relación y la explicación de tipos de datos completa se puede consultar en la recomendación del W3C *XML Schema Part 2: Datatypes,* en la dirección web https://www.w3.org/TR/xmlschema-2/.

© Ediciones Paraninfo

Comentarios en XML Schema

Existe un conjunto de elementos dentro de XML Schema que están orientados a incluir documentación que proporcione información legible por las personas y las máquinas.

El elemento **xs:annotation** permite agregar comentarios a los esquemas. Este elemento se puede agregar en cualquier lugar del documento y puede ser contenedor de los elementos **xs:documentation** y **xs:appinfo**, además de permitir cualquier otro tipo de contenido, incluido HTML.

En el siguiente fragmento de código XSD se puede observar un comentario en el que se proporciona información general sobre el validador.

```
<xs:element name="mensaje">
    <xs:annotation>
        <xs:appinfo>Validador de mensajes</xs:appinfo>
        <xs:documentation xml:lang="es">
        Este esquema valida mensajes privados seguros.
        </xs:documentation>
    </xs:annotation>
    <xs:complexType>
        <xs:sequence>
            <xs:element name="de" type="xs:string"/>
            <xs:element name="para" type="xs:string"/>
```

Recuerda

Aunque parezca paradójico que XML sea el lenguaje en el que se define la estructura de un documento escrito en XML, esto no hace sino confirmar la versatilidad y la capacidad de este lenguaje.

Actividad resuelta 4.3

Validación con XML Schema

Validar utilizando XML Schema un documento XML que contiene un registro de temperatura con la siguiente estructura:

- El elemento principal contiene un registro de temperaturas con el nombre de la provincia, la fecha del registro, la temperatura mínima y máxima y una relación de incidencias.
- La fecha de registro debe ser una fecha válida.
- La temperatura mínima no debe ser inferior a –50 °C.
- La temperatura máxima no debe ser superior a 50 °C.
- El número de incidencias no tiene límite.
- Las incidencias tienen un nombre que puede ser “frio”, “nieve”, “lluvia” o “calor”.
- Las incidencias tienen una severidad que puede ser “alta”, “media” o “baja”.

© Ediciones Paraninfo

```
<?xml version="1.0" encoding="UTF-8"?>
<registro_temperatura
    xmlns:xs="http://www.w3.org/2001/XMLSchema-instance"
    xs:noNamespaceSchemaLocation="actividad_resuelta_4_3.xsd">
    <provincia>Zamora</provincia>
    <fecha>2022-01-07</fecha>
    <temperatura_minima>-5</temperatura_minima>
    <temperatura_maxima>4</temperatura_maxima>
    <incidencias>
        <incidencia nombre="frio" severidad="alta"/>
        <incidencia nombre="nieve" severidad="alta"/>
        <incidencia nombre="lluvia" severidad="media"/>
    </incidencias>
</registro_temperatura>
```

Solución

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
<xs:simpleType name="nombre_incidencia">
    <xs:restriction base="xs:string">
        <xs:enumeration value="frio" />
        <xs:enumeration value="nieve" />
        <xs:enumeration value="lluvia" />
        <xs:enumeration value="calor" />
    </xs:restriction>
</xs:simpleType>
<xs:simpleType name="severidad_incidencia">
    <xs:restriction base="xs:string">
        <xs:enumeration value="baja" />
        <xs:enumeration value="media" />
        <xs:enumeration value="alta" />
    </xs:restriction>
</xs:simpleType>
<xs:element name="registro_temperatura">
    <xs:complexType>
        <xs:sequence>
            <xs:element name="provincia" type="xs:string"/>
            <xs:element name="fecha" type="xs:date"/>
            <xs:element name="temperatura_minima">
                <xs:simpleType>
                    <xs:restriction base="xs:integer">
                        <xs:minInclusive value="-50"/>
                    </xs:restriction>
                </xs:simpleType>
            </xs:element>
           <xs:element name="temperatura_maxima">
                <xs:simpleType>
                    <xs:restriction base="xs:integer">
                        <xs:maxInclusive value="50"/>
                    </xs:restriction>
                </xs:simpleType>
            </xs:element>
           <xs:element name="incidencias">
                <xs:complexType>
                   <xs:sequence>
```

© Ediciones Paraninfo

```
                        <xs:element name="incidencia" maxOccurs="unbounded">
                            <xs:complexType>
                                <xs:attribute name="nombre" type="nombre_
incidencia"/>
                                <xs:attribute name="severidad" type="severidad_
incidencia"/>
                            </xs:complexType>
                        </xs:element>
                    </xs:sequence>
                </xs:complexType>
            </xs:element>
        </xs:sequence>
    </xs:complexType>
</xs:element>
</xs:schema>
```

Actividad propuesta 4.4

Validación de expedientes académicos con XML Schema

Crea un validador utilizando XML Schema para un XML que almacena los expedientes académicos de un conjunto de estudiantes.

La estructura del documento XML es la siguiente:

- Un documento contiene un conjunto de expedientes.
- Cada expediente contiene el nombre de la titulación, el nombre del estudiante y un conjunto de módulos por cada estudiante.
- Cada módulo tiene un atributo con el nombre, otro atributo que indica si el módulo ya ha sido aprobado y un tercer atributo que informa del curso al que pertenece el módulo y que puede tomar los valores "Primero" o "Segundo".

A continuación se muestra un XML de ejemplo para poder observar la estructura:

```
<?xml version="1.0" encoding="UTF-8"?>
<expedientes>
    <expediente>
        <titulacion>DAM</titulacion>
        <estudiante>Ainhoa Gárate Lizarraga</estudiante>
        <modulos>
            <modulo nombre="Sistemas Informáticos" aprobado="true"
curso="Primero"/>
            <modulo nombre="Lenguajes de Marcas y Sistemas de Gestión de la
Información" aprobado="true" curso="Primero"/>
            <modulo nombre="Programación multimedia y dispositivos móviles"
aprobado="false" curso="Segundo"/>
        </modulos>
    </expediente>
    <expediente>
        <titulacion>DAW</titulacion>
        <estudiante>Ferran Soler Puig</estudiante>
        <modulos>
           <modulo nombre="Programación" aprobado="true" curso="Primero"/>
            <modulo nombre="Desarrollo web en entorno cliente" aprobado="false"
curso="Segundo"/>
```


© Ediciones Paraninfo

```
            <modulo nombre="Desarrollo web en entorno servidor" aprobado="false"
curso="Segundo"/>
        </modulos>
    </expediente>
</expedientes>
```

4.3.4. Otras formas de validación

Además de los DTD y XML Schema, existen tecnologías alternativas que tratan de cubrir las limitaciones o inconvenientes de los sistemas de validación recomendados por el W3C. Por ejemplo, XML Schema es alabado por ser robusto al estar diseñado en XML, pero criticado por su complejidad, al considerarse que tiene demasiado alcance para las necesidades cotidianas, lo que complica la creación de validadores.

Las alternativas más conocidas son RELAX NG y Schematron.

RELAX NG *(REgular LAnguage for XML Next Generation)* fue definido por OASIS, una organización sin ánimo de lucro que trabaja en estándares de seguridad, tecnologías relacionadas con los datos o computación en la nube, además de otras disciplinas. Dispone de una sintaxis XML y de otra más compacta no XML.

Schematron, por su parte, está basado en reglas y se expresa en XML. Es más potente que XML Schema y DTD, ya que contempla algunas restricciones que estos no permiten.

4.4. XML aplicado

XML es un metalenguaje polivalente y son muchos los campos en los que se puede aplicar para resolver un buen número de problemas. Estos problemas se pueden resolver utilizando otros lenguajes diseñados a medida, pero no se dispondría de la robustez de XML, de las herramientas de procesado, edición, consulta y validación (DTD y XML Schema), así como de las librerías existentes en prácticamente todos los lenguajes de programación.

4.4.1. XML en aplicaciones

Un documento XML representa, en esencia, una estructura de datos, y esta es un componente fundamental de las aplicaciones y programas informáticos. Por lo tanto, cualquier estructura de datos jerárquica puede representarse y almacenarse en forma de XML. Dado que ya están disponibles todas las herramientas software en forma de librerías para el manejo de XML, su elección permitirá centrar los esfuerzos en el diseño de la estructura de datos en lugar de en crear los mecanismos para su gestión.

La aplicación de XML en aplicaciones puede ser muy diversa: los datos de configuración de una aplicación en los que se almacenan los colores del texto o el idioma en el que se van a presentar los textos; los perfiles de seguridad de un sistema, recogiendo los diferentes perfiles junto con información referente a qué acciones pueden realizar; las

© Ediciones Paraninfo

preferencias de usuario de un videojuego, guardando la información referente al nivel de dificultad en el que quiere jugar o el nombre de su personaje. Estas son solo algunas ideas de cómo aprovechar XML para representar información en una aplicación.

Por ejemplo, una máquina de café programable que permita configurar temperatura, tipo de grano y horario de preparación para cada usuario, podría almacenar la configuración en un documento XML similar al que se muestra a continuación:

```
<?xml version="1.0" encoding="UTF-8"?>
<config>
  <coffee usuario="Fernando">
    <variedad>Arábica</variedad>
    <temperatura>61</temperatura>
    <hora>16</hora>
  </coffee>
  <coffee usuario="Rosalía">
    <variedad>Robusta</variedad>
    <temperatura>62</temperatura>
    <hora>11</hora>
  </coffee>
</config>
```

Evidentemente se podría almacenar la misma configuración en un formato creado a medida, pero sería como reinventar la rueda: si la solución existe y es buena, es mejor centrar el esfuerzo en otros problemas para los que aún no hay solución.

XML también se utiliza con frecuencia como formato de intercambio entre aplicaciones, proporcionando importantes ventajas sobre otros formatos.

Es habitual que en las empresas convivan diferentes aplicaciones programadas en distintos lenguajes y ejecutadas en diferentes ordenadores. Estas aplicaciones frecuentemente deben intercambiar información y es necesario definir un formato para que las dos partes de la comunicación se puedan entender. XML es una solución perfecta para este problema, ya que, además del formato, proporciona las herramientas de validación suficientes para garantizar que el envío y recepción de los datos se realiza correctamente.

Un **servicio web** es una aplicación de estos mecanismos de intercambio en un contexto de tecnologías y protocolos concretos.

4.4.2. XML en la web

XML sirve para diseñar páginas web, al igual que se hace con HTML. Por tanto, con XHTML se pueden crear páginas web. XHTML no es sino una versión XML de HTML, incluyendo las mismas posibilidades, pero con la robustez y fiabilidad de XML.

La otra alternativa consiste en utilizar transformaciones XSLT para convertir un documento XML en una página web.

© Ediciones Paraninfo

Sin entrar en detalles, ya que esta herramienta se trata en la Unidad 6 de este libro, con XSLT se pueden definir transformaciones que conviertan las partes de un documento XML en una salida de cualquier tipo, incluido HTML o XHTML.

Por ejemplo, dado el siguiente elemento XML:

```
<autor>Fernando Paniagua</autor>
<titulo>La Reina de los Muertos</titulo>
```

mediante una regla XSLT se puede transformar, por ejemplo, en el siguiente código HTML, creando una representación arbitraria en función de la decisión del diseñador:

```
<h1 class="autor">Fernando Paniagua</h1>
<h2 class="titulo">La Reina de los Muertos</h2>
```

Esta conversión de datos en vista mediante reglas es muy potente y la soporta la mayoría de los navegadores.

4.4.3. XML en servicios web

Un servicio web (*Web Service,* en inglés, también conocido como WS) es una tecnología que, a través de determinados protocolos y estándares, permite intercambiar datos entre sistemas informáticos.

Cuando se diseña un WS, el protocolo de comunicación de nivel de aplicación que se va a utilizar ya está decidido: HTTP, el protocolo de la web que, como su propio nombre indica, está orientado a la transferencia de hipertexto. Sobre este protocolo se puede enviar, por lo tanto, texto plano, lo que tiene que hacer pensar en XML como lenguaje para el envío y recepción de información.

Existen dos tipos de servicios web: SOAP *(Simple Object Access Protocol)* y REST *(Representational State Transfer).* Ambos tipos se basan en XML.

SOAP es un estándar muy completo y bastante complejo en el que XML es fundamental; todo en SOAP es XML, desde los propios mensajes, pasando por la definición de las estructuras de datos, hasta los esquemas de seguridad.

REST, por su parte, es mucho más sencillo de utilizar, menos robusto y más flexible. En un servicio web REST, XML es una opción. Se pueden utilizar otros lenguajes de representación de datos, frecuentemente JSON, y no es extraño que en muchos casos se proporcionen las dos opciones (XML y JSON) tanto para el envío como para la recepción de datos.

En el diseño de un servicio web, por lo tanto, hay que tomar una primera decisión relativa a que el tipo sea SOAP o REST. Esta decisión está sujeta a múltiples variables que tienen que ver con las aplicaciones que se van a comunicar. Si se elige SOAP, XML está implícito en la decisión. Si, por el contrario, se elige REST, hay que tomar una segunda decisión sobre si se va a utilizar XML o JSON como lenguaje de intercambio de datos. Esta decisión dependerá de lo robusta que deba ser la comunicación, ya que JSON, aunque más sencillo y ligero, no dispone de las herramientas de XML.

© Ediciones Paraninfo

4.5. Herramientas de visualización y edición de XML

Las herramientas que se pueden utilizar para trabajar con XML dependen de las necesidades y del uso que se vaya a hacer de los documentos. Un documento XML está formado por texto plano, sin añadidos, por lo que cualquier editor de texto puede servir para crearlo o modificarlo. Lógicamente, un editor muy simple que no proporcione ningún tipo de ayuda (corrector sintáctico, código de colores, etc.) hará que el trabajo sea más difícil, por lo que desearemos utilizar herramientas más potentes capaces de prestar una ayuda bien recibida.

Si lo que se quiere es consultar un documento, también se podrá utilizar un editor de texto plano, pero, al igual que pasa en la edición, se echará de menos algún tipo de ayuda visual en forma de representación en forma de árbol, código de colores, etc., por lo que será interesante disponer de editores o herramientas más sofisticadas.

Por último, puede que se desee realizar algún tipo de proceso sobre un documento XML, como por ejemplo la validación con un DTD o un XML Schema. En este caso no hay opción: o la herramienta hace el trabajo, o no sirve. Es posible validar «a ojo» un documento XML con DTD o XML Schema, pero no es recomendable porque resulta demasiado sencillo cometer errores.

4.5.1. Editores de texto

Al hablar de editores de texto se está haciendo referencia a un abanico de programas que van desde aquellos con una funcionalidad básica, como el Bloc de Notas de Windows, hasta otros más complejos, como Notepad++ o Visual Studio Code, por citar algunos ejemplos.

Los más sencillos permiten tanto consultar como editar los ficheros, pero no proporcionan ningún tipo de ayuda específica referente al manejo de XML.

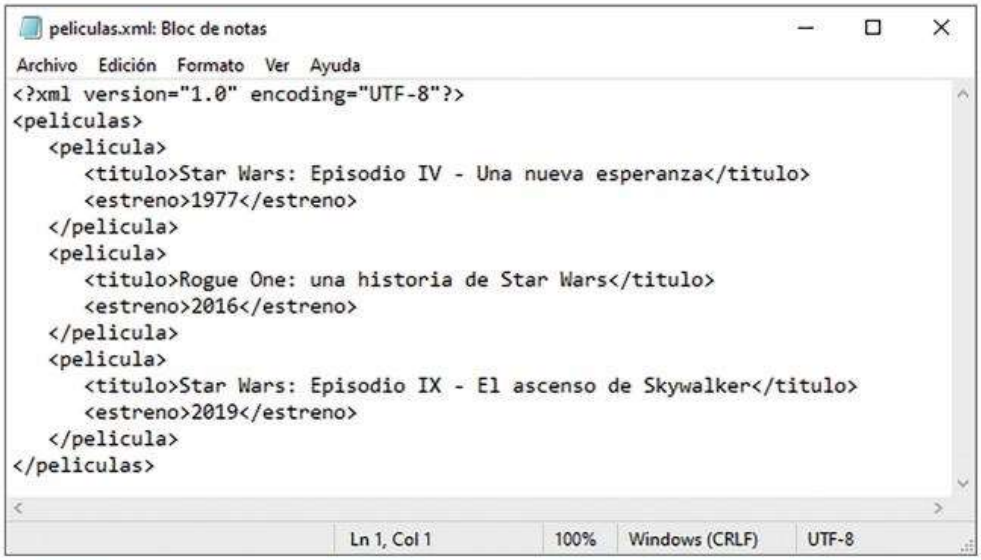


Figura 4.13. *Los editores más básicos no proporcionan ningún tipo de ayuda.*

© Ediciones Paraninfo

Los editores más complejos suelen reconocer la gramática y la sintaxis de XML y ayudan mediante el uso de esquemas de colores o permitiendo contraer o extender los elementos para facilitar la consulta de estos.

```
<?xml version="1.0" encoding="UTF-8"?>
<peliculas>
    <pelicula>
        <titulo>Star Wars: Episodio IV - Una nueva esperanza</titulo>
        <estreno>1977</estreno>
    </pelicula>
    <pelicula>
    <pelicula>
        <titulo>Star Wars: Episodio IX - El ascenso de Skywalker</titulo>
        <estreno>2019</estreno>
    </pelicula>
</peliculas>
```

Figura 4.14. *Los editores más sofisticados incluyen códigos de colores y permiten agrupar elementos.*

4.5.2. Navegadores web

Los navegadores web de uso más extendido tienen la capacidad de representar documentos XML.

Los navegadores muestran los documentos XML como un árbol en el que es posible comprimir o expandir las ramas, permitiendo tener diferentes perspectivas de la estructura y los datos. Además, suelen utilizar un código visual, presentando el contenido del documento con colores y estilos de letra diferentes según presenten elementos, atributos y sus valores o textos, facilitando la lectura del documento.

En la Figura 4.15 se muestra un documento XML con la predicción meteorológica para la ciudad de Aranjuez proporcionado por la Agencia Estatal de Meteorología. Como se puede observar, las líneas están sangradas y los elementos agrupados según su jerarquía.

```
This XML file does not appear to have any style information associated with it. The document tree is shown below.

▼<root xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 version="1.0" xsi:noNamespaceSchemaLocation="http://www.aemet.es/xsd/localidades.xsd">
  ▼<origen>
     <productor>Agencia Estatal de Meteorología - AEMET. Gobierno de España</productor>
     <web>http://www.aemet.es</web>
     <enlace>http://www.aemet.es/es/eltiempo/prediccion/municipios/aranjuez-id28013</enlace>
     <language>es</language>
     <copyright>© AEMET. Autorizado el uso de la información y su reproducción citando a AEMET como autora
     </copyright>
     <nota_legal>http://www.aemet.es/es/nota_legal</nota_legal>
   </origen>
   <elaborado>2020-10-02T19:09:49</elaborado>
   <nombre>Aranjuez</nombre>
   <provincia>Madrid</provincia>
  ▼<prediccion>
```

Figura 4.15. *Documento XML proporcionado por la Agencia Estatal de Meteorología.*

© Ediciones Paraninfo

Los navegadores también pueden interpretar dialectos de XML, como por ejemplo el formato de representación de gráficos vectoriales SVG. Al abrir un fichero SVG, el navegador interpretará el código y representará la imagen descrita en él, tal y como se puede ver en la Figura 4.16.

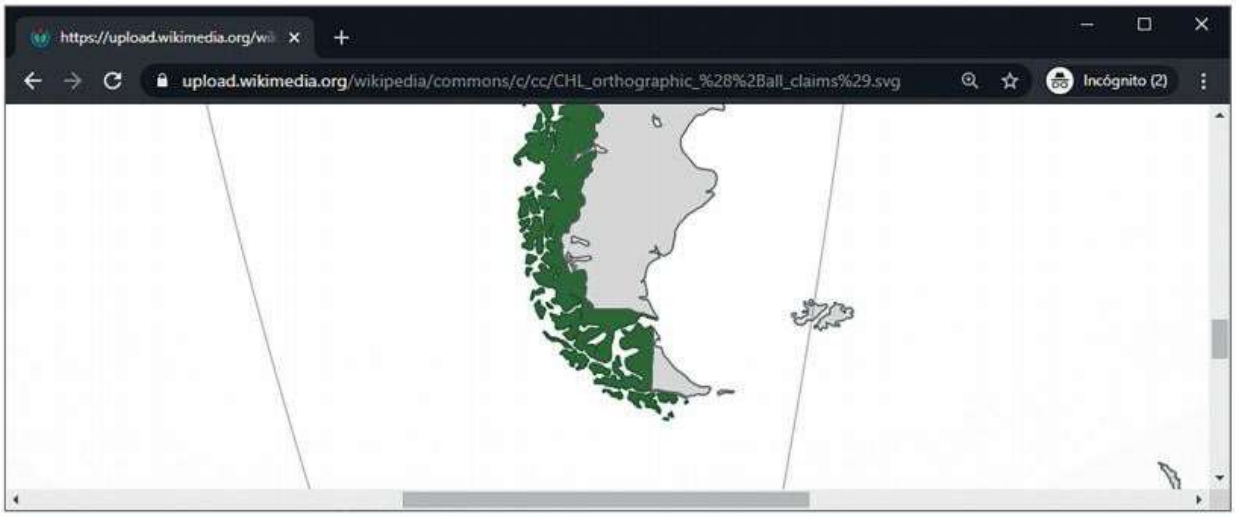


Figura 4.16. *Representación visual de un fichero gráfico SVG en Google Chrome.*

4.5.3. XML Notepad

Este sencillo editor permite consultar, crear y modificar documentos XML de forma visual. Se encuentra disponible en la cuenta de GitHub de Microsoft, es gratuito y muy sencillo de utilizar.

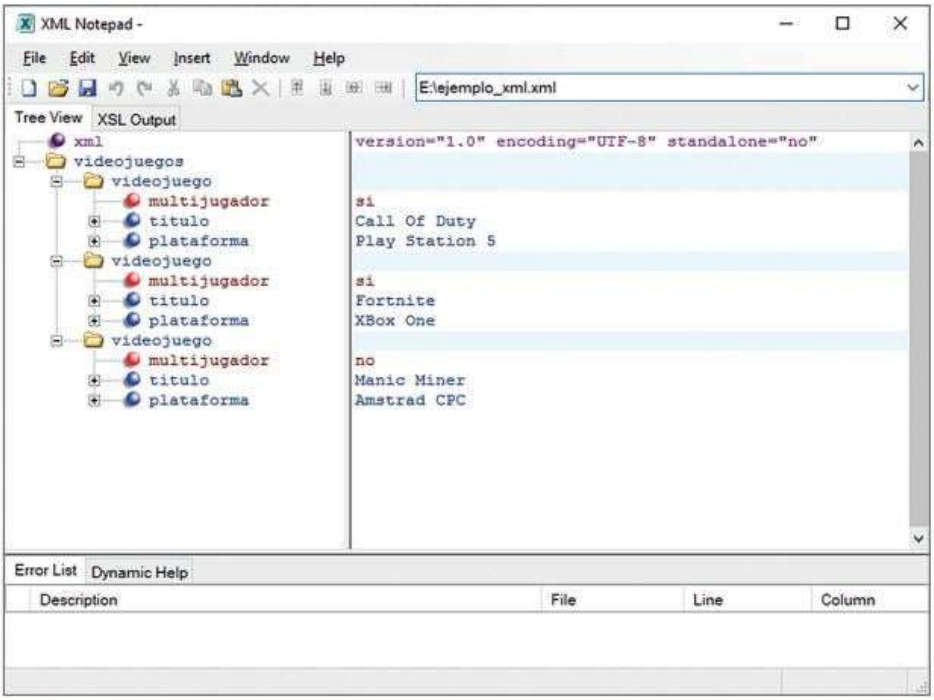


Figura 4.17. *XML Notepad es un editor sencillo y ligero orientado a XML.*

© Ediciones Paraninfo

La presentación de documentos XML en esta herramienta es muy gráfica: muestra la estructura de un árbol, permitiendo desplegar o contraer los elementos contenedores; asigna unos iconos de distinto color a los atributos y a los elementos; muestra los valores de los atributos en un color y los textos contenidos en los elementos en otro.

XML Notepad también procesa el documento y valida que está bien formado, advirtiendo si encuentra algún error.

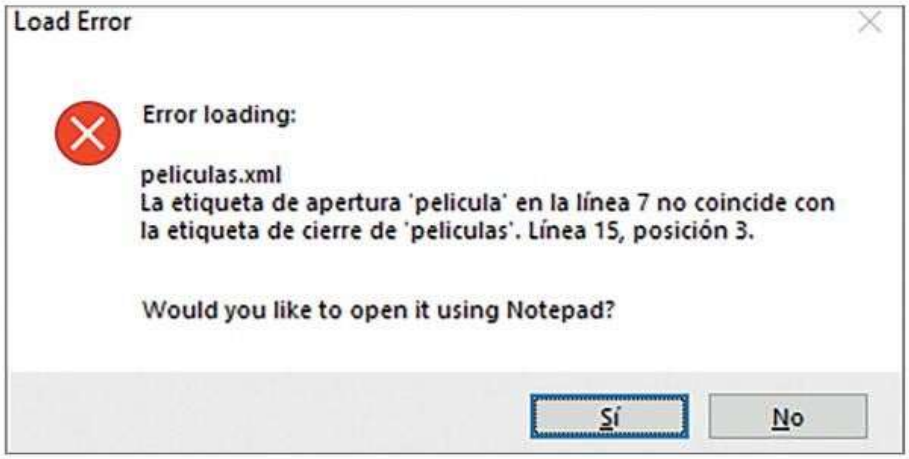


Figura 4.18. *Los mensajes de error incluyen el número de línea y la posición de este.*

Desde el punto de vista de la edición, permite insertar elementos, atributos y demás contenidos desde una interfaz gráfica fácil e intuitiva.

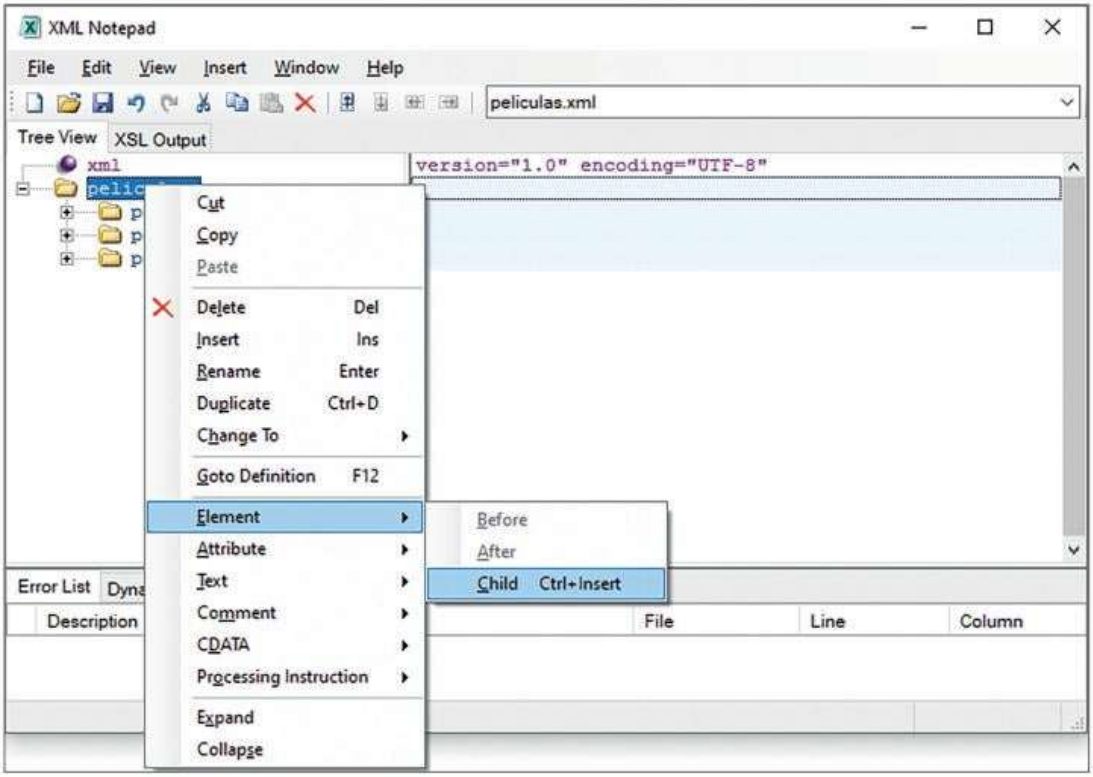


Figura 4.19. *Los editores orientados a XML incorporan funcionalidad específica de este lenguaje.*

© Ediciones Paraninfo

4.5.4. Liquid Studio

Liquid Studio es un potente editor que permite trabajar XML, además de con otros tipos de ficheros. Dispone de interfaz gráfica, lo que facilita el mantenimiento de los documentos. Comprueba la sintaxis y valida con DTD y con XML Schema, proporcionando información detallada relativa a los errores, lo que convierte a este editor en una herramienta muy útil para el mantenimiento de los documentos XML.

Es un producto comercial, pero existe una versión gratuita para uso personal.

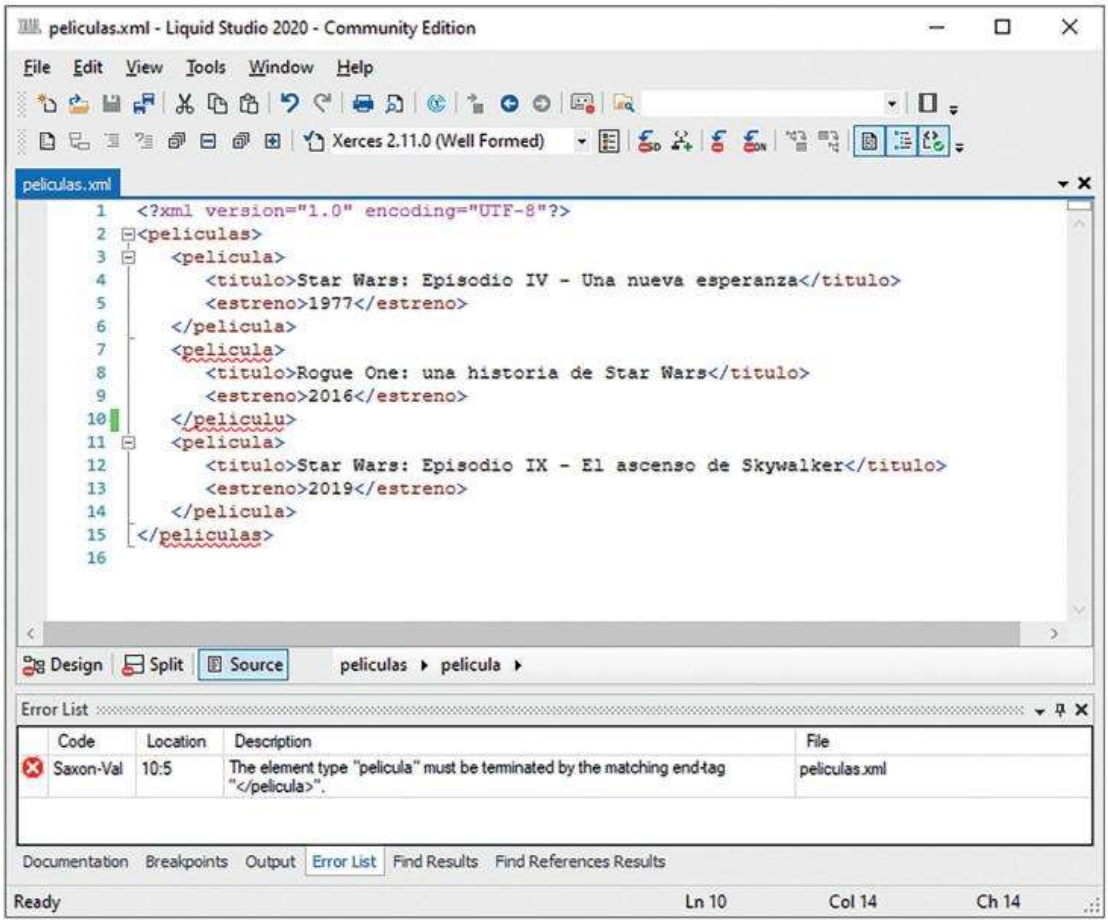


Figura 4.20. *La versión gratuita de Liquid Studio no tiene disponible toda la funcionalidad de la versión comercial.*

4.5.5. XMLPad

XmlPad es un editor de documentos XML para dispositivos móviles de la marca Apple. Se puede conseguir a través de la tienda de aplicaciones de este fabricante (APP Store). También dispone de una versión para Windows, conocida como WmHelp XmlPad.

WmHelp XmlPad es un editor gratuito que permite consultar y editar documentos XML, darles formato, así como validarlos con DTD, RELAX NG y Schematron.

© Ediciones Paraninfo

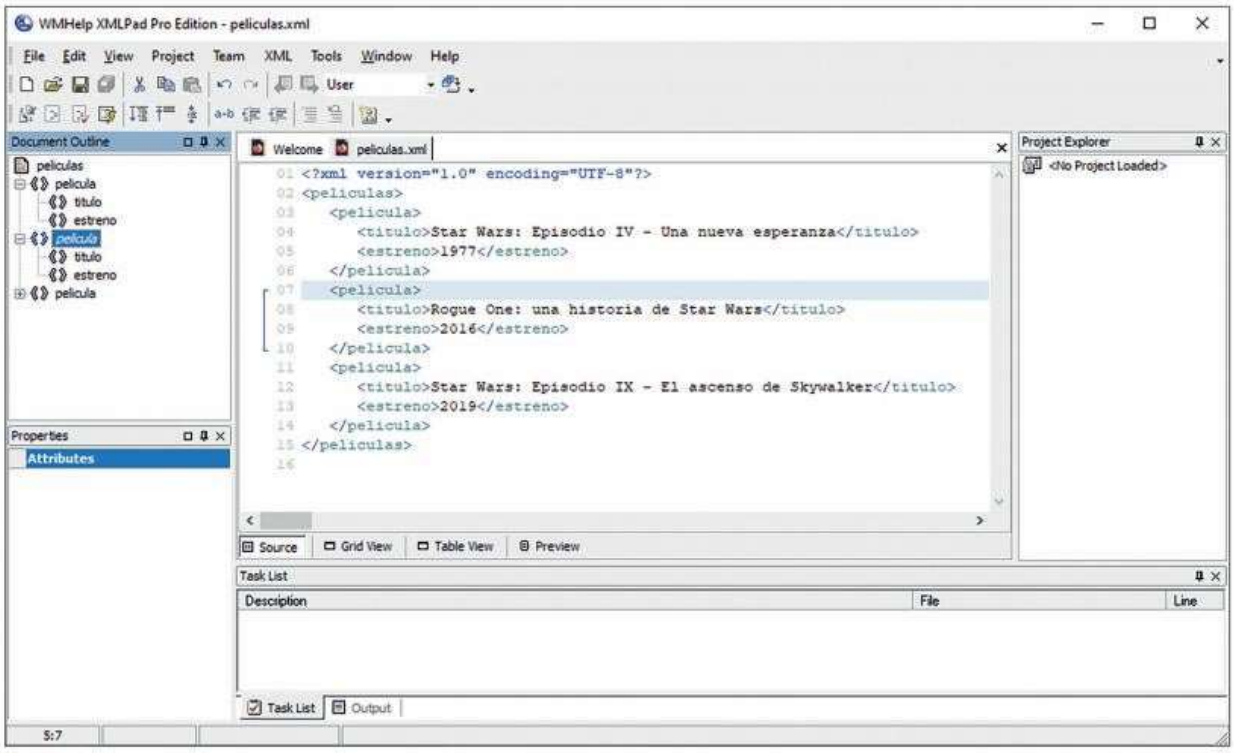


Figura 4.21. *XMLPad Pro es una herramienta muy intuitiva.*

También ofrece múltiples perspectivas de los elementos, facilitando la consulta de los documentos cuando estos son complejos.

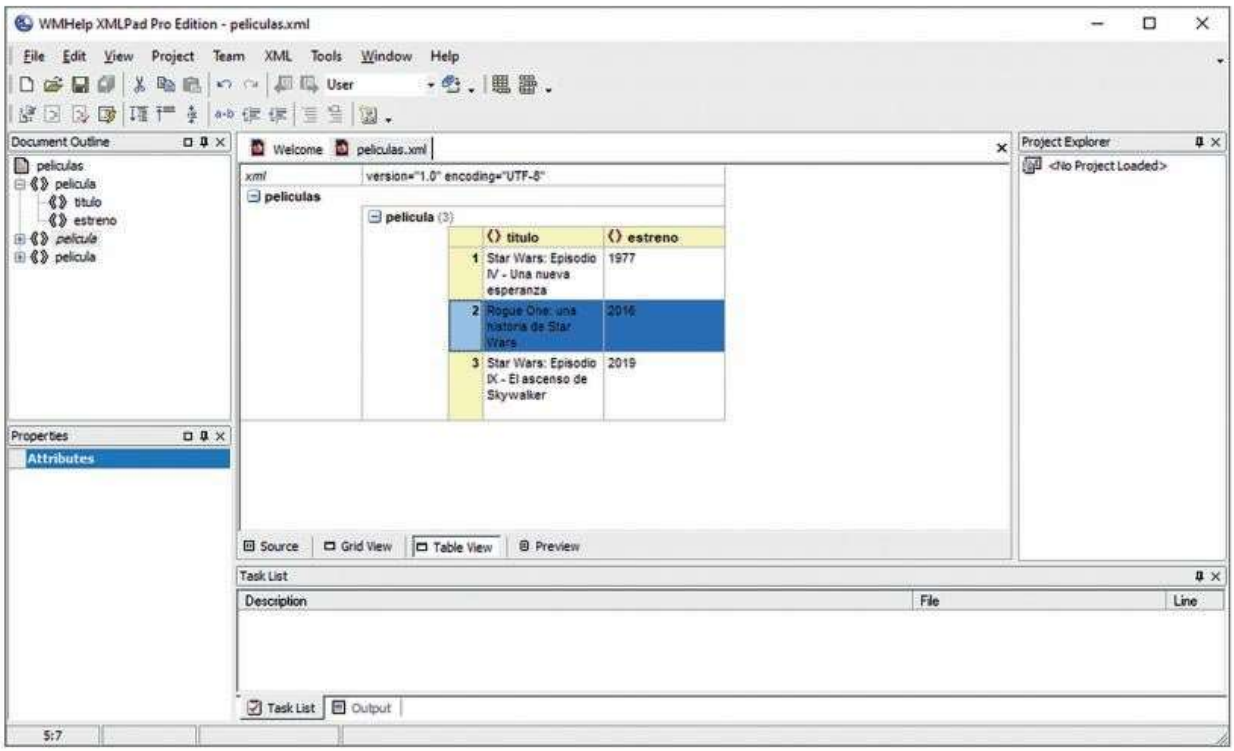


Figura 4.22. *Las vistas gráficas facilitan la lectura de los documentos.*

© Ediciones Paraninfo

4.5.6. Eclipse

Eclipse es un famoso entorno de desarrollo integrado (IDE, *Integrated Development Environment*) que se utiliza para programar aplicaciones en diversos lenguajes. Dispone de un repositorio llamado *Eclipse Marketplace,* que contiene extensiones para realizar una gran diversidad de tareas, incluida la gestión de documentos HTML, XML, CSS, etcétera.

Por medio de estas ampliaciones, concretamente con el componente Eclipse XML Editors and Tools, Eclipse se convierte en una herramienta muy completa para la edición de documentos XML.

Nota técnica

La gran mayoría de editores de texto, de código y de IDE (entornos de desarrollo integrados) disponen de soporte para la edición de documentos basados en lenguajes de marcas como XML y HTML. Antes de cambiar de editor, puede resultar interesante buscar una extensión para la herramienta con la que se está trabajando para la gestión de documentos basados en lenguajes de marcas.

Figura 4.23. *Eclipse es un IDE muy versátil gracias a su capacidad de ampliación.*

© Ediciones Paraninfo

4.5.7. Editores web

Existe un buen número de editores online a los que se puede acceder a través del navegador web. La principal ventaja consiste en que no es necesario realizar ninguna instalación.

Algunos de estos editores son XML Viewer, de la web Code Beauty; XML Editor, de la web JSON formatter, y Online XML Editor, de la web Tutorial Point.

Suelen incorporar codificación con colores, sangrado automático, representación en forma de árbol, comprobación de sintaxis y validación con DTD y XML Schema en muchas ocasiones.

Los editores web representan una alternativa sencilla y de acceso rápido, suficiente para satisfacer las necesidades básicas.

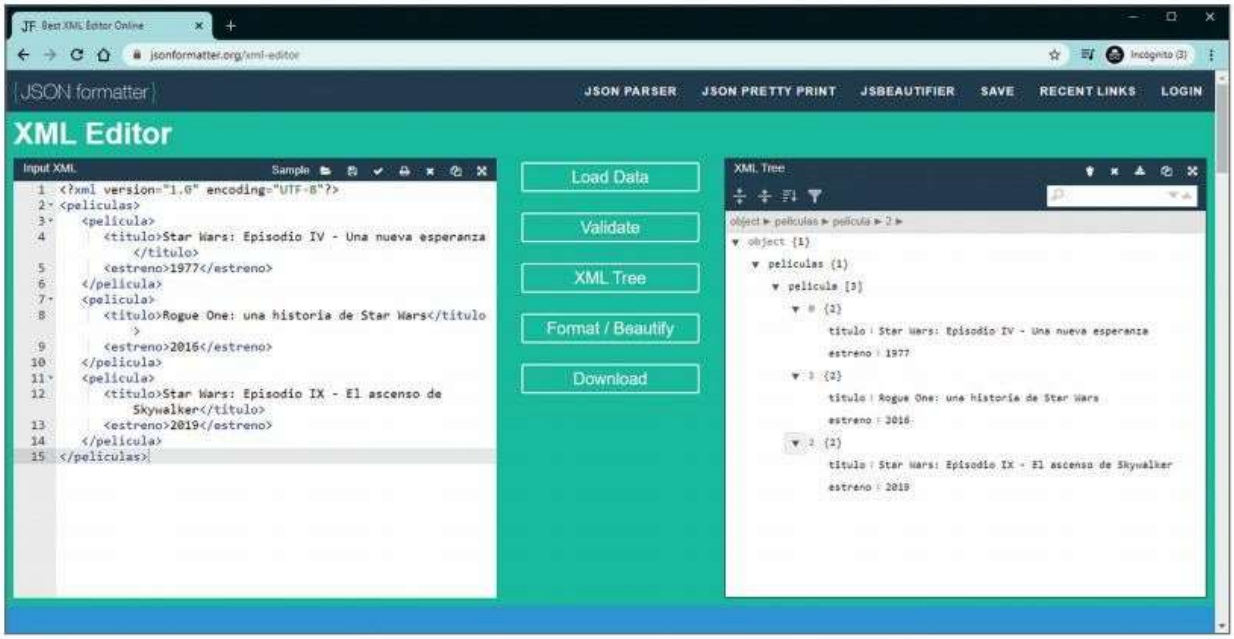
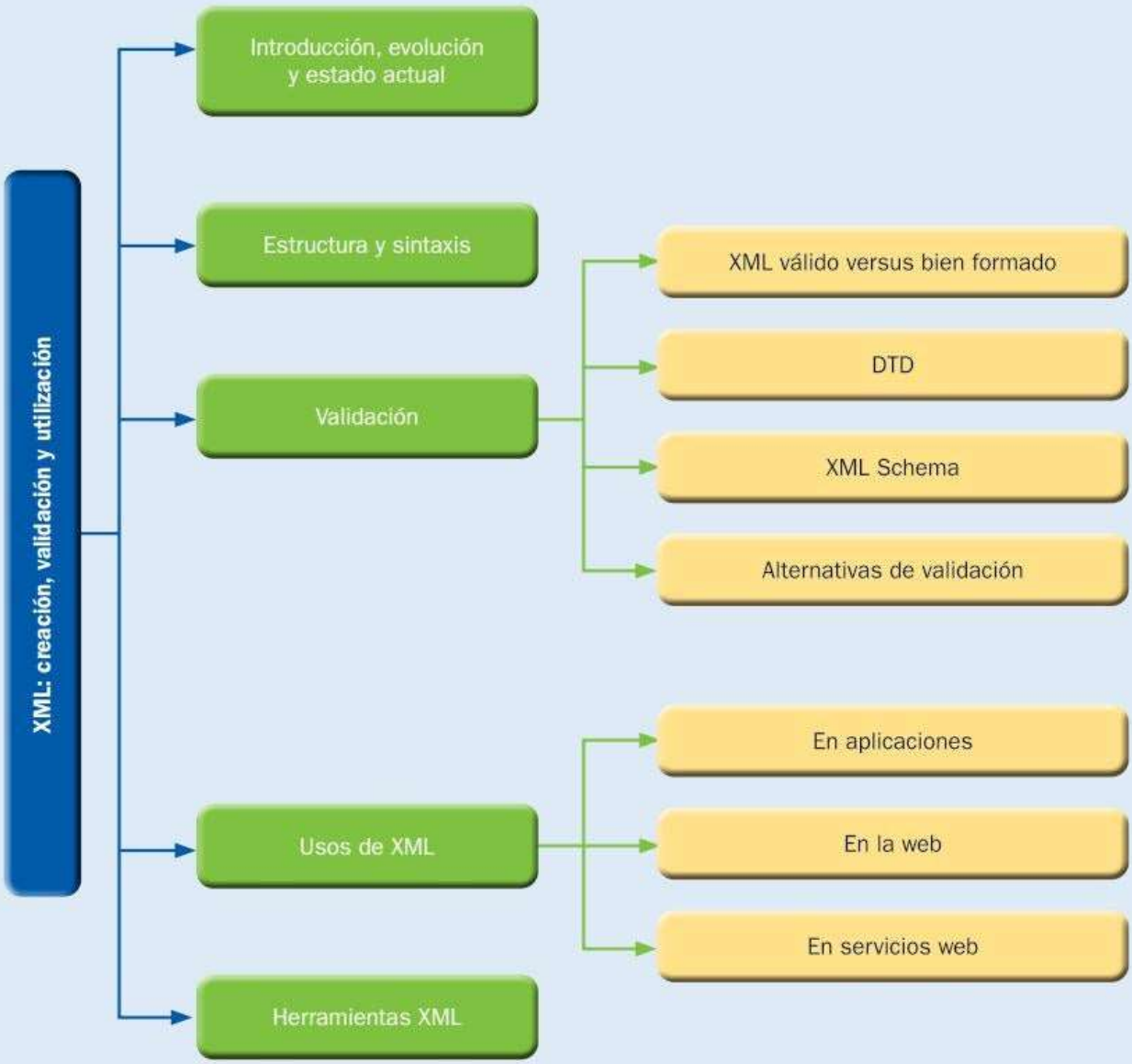


Figura 4.24. *Los editores web o editores en línea son una alternativa para la edición de documentos XML.*

© Ediciones Paraninfo



© Ediciones Paraninfo

Actividades de comprobación

4.1. Indica qué afirmaciones son correctas respecto a los nombres de elementos en XML:

a) Diferencian entre mayúsculas y minúsculas.
b) Deben comenzar por una letra o un guion bajo.
c) Las etiquetas de apertura y cierre deben ser iguales.
d) Todas las afirmaciones son correctas.

4.2. Indica cuál de las siguientes opciones es un comentario en XML:

a) <?-- Comentario -->
b) <!-- Comentario -->
c) //Comentario
d) /* Comentario */

4.3. ¿Cuál de las siguientes declaraciones se corresponde con la entidad «mayor que»?

a) <
b) >
c) '
d) "

4.4. ¿Cuál de las siguientes opciones se corresponde con la declaración de una sección CDATA?

a) <CDATA contendio >
b) <CDATA[contenido]>
c) <![CDATA[contenido]]>
d) <CDATA![contenido]>

4.5. Cuando un documento XML es conforme a la sintaxis de XML, se dice que es:

a) Válido.
b) Correcto.
c) Bien formado.
d) Conforme.

4.6. ¿Cómo se hace referencia a la entidad *nombre-actor* en un DTD?

a) !nombre-actor;
b) @nombre-actor;
c) &nombre-actor;
d) #nombre-actor;

4.7. ¿Qué palabra indica que un elemento debe estar vacío en un DTD?

a) CLEAR
b) EMPTY
c) ANY
d) (#PCDATA)

4.8. ¿Cómo se indica en un DTD que elemento1 contiene uno y solo un elemento2?

a) <!ELEMENT elemento1 (elemento2)>
b) <!ELEMENT elemento1 (elemento2?)>
c) <!ELEMENT elemento1 (elemento2+)>
d) <!ELEMENT elemento1 (elemento2*)>

© Ediciones Paraninfo

4.9. ¿Cómo se indica en un DTD que elemento1 contiene cero y más elementos2?

a) <!ELEMENT elemento1 (elemento2)>
b) <!ELEMENT elemento1 (elemento2?)>
c) <!ELEMENT elemento1 (elemento2+)>
d) <!ELEMENT elemento1 (elemento2*)>

4.10. ¿Cómo se indica en un DTD que elemento1 puede contener texto?

a) <!ELEMENT elemento1 (text)>
b) <!ELEMENT elemento1 (DATA)>
c) <!ELEMENT elemento1 (#PCDATA)>
d) <!ELEMENT elemento1 (string)>

4.11. ¿Qué elemento se usa en un DTD para declarar un atributo?

a) !DOCTYPE
b) !ELEMENT
c) !ATTLIST
d) !NOTATION

4.12. ¿Qué palabra clave indica que un atributo es opcional?

a) #REQUIRED
b) #IMPLIED
c) #FIXED
d) #OPTIONAL

4.13. ¿Cómo se indica en XML Schema el número mínimo de veces que debe aparecer un elemento?

a) Con la propiedad numMin.
b) Con la propiedad minOccurs.
c) Con la propiedad min.
d) XML Schema no permite indicar el número mínimo de veces que aparece un elemento.

4.14. ¿Qué etiqueta se corresponde con la declaración de un atributo en XML Schema?

a) xs:label
b) xs:element
c) xs:attribute
d) xs:attr

4.15. ¿Qué etiqueta hay que utilizar en XML Schema para indicar que un elemento debe contener obligatoriamente un elemento de una lista?

a) <xs:sequence>
b) <xs:choise>
c) <xs:all>
d) <xs:list>

4.16. ¿Cómo se indica en XML Schema que el contenido de un elemento debe tener una longitud mínima?

a) xs:minLength
b) xs:minExclusive
c) xs:minInclusive
d) xs:length

© Ediciones Paraninfo

4.17. ¿Cuál es el tipo de dato que se utiliza en XML Schema para indicar que un elemento debe contener un número entero?

a) xs:string
b) xs:integer
c) xs:decimal
d) xs:boolean

Actividades de aplicación

4.18. Crea un documento XML que represente un billete de avión. La información contenida es: nombre del pasajero, aeropuerto de origen, aeropuerto de destino, fecha y hora de vuelo. No utilices atributos, solo elementos.

4.19. Efectúa un DTD interno que valide el formato del documento XML de la Actividad de aplicación 4.18.

4.20. Crea un documento XML que represente un pedido a una hamburguesería. La información contenida es: nombre del cliente, dirección de entrega, una lista de productos que incluya el nombre y el precio. Por cada producto, incluye un atributo que contenga el número de productos.

4.21. Realiza un DTD externo que valide el formato del documento XML de la Actividad de aplicación 4.20.

4.22. Crea un documento XML que represente una factura. La información contenida es: número, fecha, nombre y dirección del cliente, listado de productos que incluya nombre, número de unidades y precio unitario, subtotal, IVA aplicado y total. Utiliza elementos o atributos según consideres oportuno.

4.23. Realiza un documento XML que contenga tu ficha personal. Decide tú mismo qué estructura, elementos y atributos utilizar.

4.24. Crea un DTD que valide un documento XML con la siguiente estructura: un elemento principal representando una película que contiene obligatoriamente un título, un director, una sinopsis, uno o más actores y, opcionalmente, un año de estreno.

4.25. Realiza un DTD que valide un documento XML con la siguiente estructura: un elemento principal representando un videojuego que contiene un atributo título obligatorio, un título obligatorio y obligatoriamente uno o más géneros.

4.26. Valida, utilizando XML Schema, un documento XML con la siguiente estructura: un elemento principal representando un libro que contenga obligatoriamente la siguiente secuencia de elementos: título, autor y género.

4.27. Verifica, utilizando XML Schema, un documento XML con la siguiente estructura: un elemento principal representando un libro que contenga obligatoriamente la siguiente secuencia de elementos: título, autor y género.

© Ediciones Paraninfo

4.28. Comprueba que es válido, utilizando XML Schema, un documento XML con la siguiente estructura: un elemento principal representando una incidencia que contenga un nombre de cliente (obligatorio), fecha y hora de notificación y un tipo (a elegir uno entre «técnica», «de facturación», «de servicio»).

4.29. Utiliza XML Schema para comprobar que un documento XML es válido dada la siguiente estructura: un elemento principal representando una serie que contenga un título, una fecha de estreno, un número de temporadas representado por un valor entero con un valor mínimo de 1 y máximo de 50 y un género que elegir entre «comedia», «drama» y «ciencia ficción».

Actividades de ampliación

4.30. Busca tres editores de XML online. Escribe el mismo documento XML en todos ellos. Después elige tu editor favorito en función de la sencillez de uso, la apariencia estética y la claridad con la que informa de los errores.

4.31. Crea un documento XML en un editor online. Escribe un documento XML correcto en dicho editor. Modifica el documento insertando errores y comprueba que el editor los detecta. A continuación observa los mensajes de error y determina si proporciona suficiente información.

4.32. Busca información referente a la comparación entre DTD y XML Schema. Profundiza en las razones por las que hay que elegir uno u otro en función de la validación que desees hacer.

4.33. Profundiza en al menos tres de los tipos de datos que admite XML Schema y que no se explican en este libro.

4.34. Halla información referente a las validaciones relacionadas con RELAX NG.

4.35. Busca información referente a las validaciones relacionadas con Schematron.

Enlaces web de interés

- **XML en w3c.es** - https://www.w3.org/XML/
  (Información oficial en inglés sobre XML)
- **XML en MDN** - https://developer.mozilla.org/es/docs/Web/XML/Introducción_a_XML
  (Información sobre XML en la web Mozilla Development Network)
- **Tutorial DTD en w3school** - https://www.w3schools.com/xml/xml_dtd_intro.asp
  (Tutorial sobre DTD en la web de w3school)

© Ediciones Paraninfo

- **Tutorial XML Schema en w3school** - https://www.w3schools.com/xml/schema_intro.asp
  (Tutorial sobre XML Schema en la web de w3school)

- **Visual Studio Code** - https://code.visualstudio.com
  (Portal web del editor Visual Studio Code)

- **XML Notepad** - https://github.com/microsoft/XmlNotepad
  (Repositorio público del proyecto del editor XML Notepad)

- **XML Pad** - https://xmlpad-mobile.com/
  (Portal web de la aplicación XML Pad)

- **Liquid Technologies** - https://www.liquid-technologies.com
  (Portal web de Liquid Technologies, empresa propietaria de la aplicación XML Studio)

- **Eclipse** - https://www.eclipse.org
  (Portal del entorno de desarrollo integrado Eclipse)

- **XML Viewer – Code Beauty** - https://codebeautify.org/xmlviewer
  (Editor de XML en la web Code Beauty)

- **XML Viewer – JSON Formatter** - https://jsonformatter.org/xml-viewer
  (Editor de XML en la web Code Beauty)

- **XML Editor – Tutorial Point** - https://www.tutorialspoint.com/online_xml_editor.htm
  (Editor de XML en la web Tutorial Point)

© Ediciones Paraninfo

UNIDAD 5

Almacenamiento de información

Objetivos

- Clasificar las distintas alternativas existentes para el almacenamiento de documentos XML.
- Conocer la lógica de los procesadores de documentos XML más habituales y aprender a integrarlos en un lenguaje de programación.
- Aprender la utilidad de las diversas tecnologías que forman parte del universo de XML.
- Exponer los lenguajes de manejo de documentos en formato XML disponibles para acceder a la información contenida en estos.
- Presentar algunas de las herramientas disponibles para la visualización y edición de documentos XML.

Contenidos

- 5.1. Creación de documentos XML desde programa
- 5.2. Sistemas de almacenamiento de documentos XML
- 5.3. Técnicas de búsqueda de información en documentos XML: lenguajes de consulta y manipulación

Introducción

El almacenamiento y la posterior lectura de ficheros XML son tareas necesarias para poder trabajar con este tipo de documentos. Existen varias alternativas para realizar estas acciones, desde las más laboriosas, basadas en la creación de programas, hasta las más rápidas y sencillas, basadas en la utilización de herramientas específicas.

Una vez resuelto el problema del almacenamiento, hay que procesar la información contenida en los documentos. Existen tecnologías y herramientas para habilitar el acceso a los datos de los documentos XML, mediante sistemas de consultas capaces de filtrar la información basándose en reglas lógicas o a través de procesos de transformación que pueden convertir los datos representados con XML a otros formatos.

En esta unidad se abordan las técnicas para almacenamiento, acceso y transformación de XML.

5.1. Creación de documentos XML desde programa

La creación de documentos XML se puede realizar de varias maneras. La más sencilla consiste en abrir un editor de texto, escribir las etiquetas y los contenidos del documento respetando la sintaxis y guardar el resultado como un fichero con extensión .xml.

Otra de las alternativas consiste en construir el documento mediante un programa escrito en cualquier lenguaje de programación que tenga soporte para XML. Java es uno de esos lenguajes.

Java dispone de una API llamada JAXP *(Java API for XML Processing),* que proporciona mecanismos para analizar, transformar, validar y consultar documentos XML. Esta API aporta independencia de la implementación concreta que se esté utilizando.

Nota técnica

API es la abreviatura de *Application Programming Interface* («interfaz de programación de aplicaciones») y en programación orientada a objetos representa un conjunto de clases y métodos proporcionados por una biblioteca de componentes software.

La API JAXP incluye un importante número de paquetes que cubren gran parte de la funcionalidad que se necesita para manejar un documento XML: mecanismos para la creación en memoria de documentos, analizadores SAX y DOM, mecanismos de validación, mecanismos de transformación y mecanismos de acceso a los datos. También implementa el estándar StAX *(Streaming API for XML),* que es un punto intermedio entre SAX y DOM.

Los paquetes incluidos en JAXP son los siguientes:

- `javax.xml.datatype`
- `javax.xml.namespace`

© Ediciones Paraninfo

- `javax.xml.parsers`
- `javax.xml.stream`
- `javax.xml.stream.events`
- `javax.xml.stream.util`
- `javax.xml.transform`
- `javax.xml.transform.dom`
- `javax.xml.transform.sax`
- `javax.xml.transform.stax`
- `javax.xml.transform.stream`
- `javax.xml.validation`
- `javax.xml.xpath`
- `org.w3c.dom`
- `org.w3c.dom.ranges`
- `org.w3c.dom.traversal`
- `org.w3c.dom.bootstrap`
- `org.w3c.dom.events`
- `org.w3c.dom.ls`
- `org.xml.sax`
- `org.xml.sax.ext`
- `org.xml.sax.helpers`

Las API de los analizadores **SAX** *(Simple API for XML)* y **DOM** *(Document Object Model)* son las más importantes para el manejo de documentos XML. Cada una de ellas tiene un funcionamiento concreto, lo que las hace adecuadas para cubrir diferentes necesidades.

SAX es un mecanismo simple y eficiente desde el punto de vista del uso de la memoria, ya que va realizando el proceso del documento XML según se realiza la lectura. La mecánica de SAX es la siguiente: el documento se lee secuencialmente y según van sucediéndose eventos (comienzo del documento, comienzo del elemento, final del elemento, final del documento, etc.), determinados métodos «callback» son invocados por el analizador. Esto implica que la lectura siempre se realiza hacia delante y que el uso de memoria es muy bajo, pues no se crea en memoria la estructura de árbol que hace el analizador DOM. SAX solo permite lectura y, por lo tanto, no será válido para crear nuevos documentos o modificar documentos existentes.

DOM, en cambio, almacena en memoria todo el documento XML. Proporciona una mayor versatilidad, puesto que va a permitir utilizar lenguajes de búsqueda y selección como XPath, así como crear documentos desde cero o modificar documentos existentes. DOM se utiliza para el almacenamiento en memoria tanto de documentos XML como de HTML y XHTML.

© Ediciones Paraninfo

Nota técnica

XPath es un lenguaje utilizado para recorrer documentos XML mediante el uso de expresiones.

En la Tabla 5.1 se comparan estos dos analizadores de XML.

Tabla 5.1. Comparativa de los analizadores SAX y DOM

Característica	SAX	DOM
Tipo de API	Tipo «push», en «streaming»	Árbol en memoria
Facilidad de uso	Media	Alta
Admite XPath	No	Sí
Consumo de recursos	Bajo	Variable, en función del documento
Solo permite lectura hacia delante	Sí	No
Permite leer XML	Sí	Sí
Permite construir XML	No	Sí
Creación, lectura, modificación y borrado de elementos	No	Sí

Previo al almacenamiento del documento XML es necesario crear el objeto en memoria para poder almacenarlo posteriormente.

El proceso de **creación** en memoria de un documento XML es sencillo, tal y como se puede observar en el siguiente bloque de código Java:

```
DocumentBuilderFactory documentFactory =
     DocumentBuilderFactory.newInstance();
DocumentBuilder documentBuilder =
     documentFactory.newDocumentBuilder();
Document document = documentBuilder.newDocument();
```

En primer lugar, se obtiene una instancia de la clase abstracta **DocumentBuilderFactory.** El objeto obtenido proporciona, a su vez, una instancia de la clase abstracta **DocumentBuilder.** Este último objeto es el que crea una implementación de la interfaz **Document,** que representa en memoria un documento HTML o XML (inicialmente vacío) sobre el que construir el documento XML final.

Una vez creado el objeto **Document,** se deben crear los contenidos del documento XML. Estos van a ser, fundamentalmente, instancias de **Element** (elementos), de **Attr** (atributos) y de **Text** (textos). Todas estas interfaces heredan de una superinterfaz **Node,** que las agrupa junto a otras de uso menos frecuente.

© Ediciones Paraninfo

En el siguiente bloque de código se muestra la creación de los distintos nodos que van a formar el documento:

```
Element elementLibro = document.createElement("libro");
Element elementTitulo = document.createElement("titulo");
Attr attributeDisponible =
     document.createAttribute("disponiblePapel");
attributeDisponible.setValue("true");
Text textTitulo =
     document.createTextNode("LENGUAJES DE MARCAS Y ...");
```

Todos los nodos se crean a partir de la instancia de **Document** obtenida inicialmente. Cada uno de estos nodos, ya sean elementos, atributos o textos, son independientes del **Document** desde el que se crearon, por lo que hay que vincularlos.

En la Figura 5.1 se muestra una representación del estado de los elementos y nodos creados a partir de la instancia de **Document.** Hasta que no se establezcan las relaciones entre ellos, son elementos independientes.

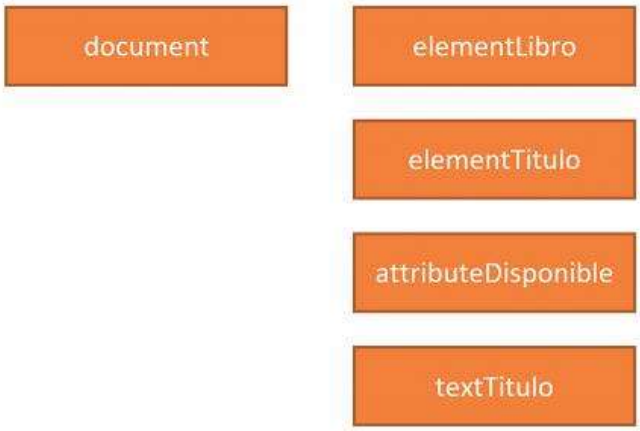


Figura 5.1. *Al crear los elementos, estos no tienen vínculos entre sí.*

La dependencia entre los distintos nodos sigue las reglas de XML: un atributo o un texto son parte de un elemento; un elemento es parte de un documento. Únicamente hay que tener cuidado a la hora de establecer la relación entre las partes para dar lugar a un documento XML bien formado de acuerdo con la estructura elegida.

```
elementTitulo.appendChild(textTitulo);
elementLibro.appendChild(elementTitulo);
elementLibro.setAttribute("disponibleDigital", "true");
elementLibro.setAttributeNode(attributeDisponible);
document.appendChild(elementLibro);
```

- En la primera línea, se está añadiendo el nodo de tipo **Text** textTitulo como hijo del nodo elementTitulo.
- En la segunda línea, se está añadiendo el nodo de tipo **Element** elementTitulo como hijo de elementLibro.

© Ediciones Paraninfo

- En la tercera línea se está asignando un atributo a elementLibro mediante un método de asignación directa.
- En la cuarta línea se está asignando el nodo de tipo **Attr** attributoDisponible a elementLibro.
- Por último, el nodo de tipo **Element** elementLibro se está añadiendo como hijo al objeto de tipo **Document** con nombre document.

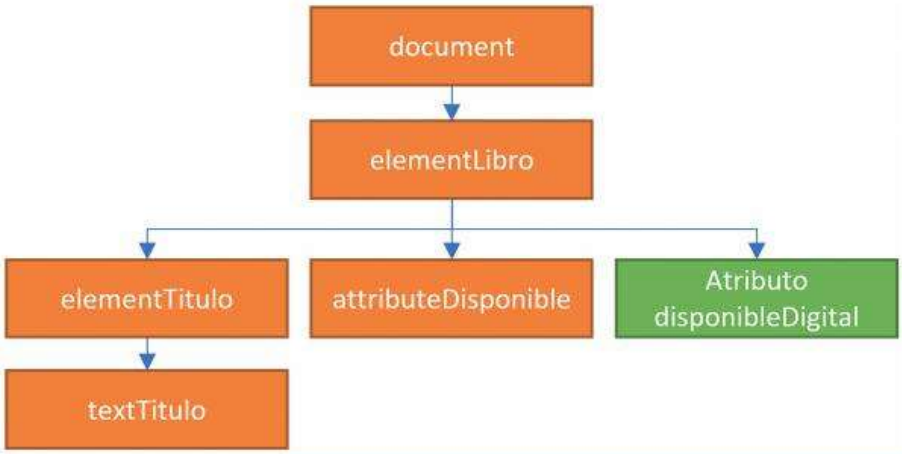


Figura 5.2. *Una vez establecidas las relaciones, el documento se convierte en una estructura de datos.*

Es importante observar que se está construyendo un árbol con dependencias jerárquicas entre los nodos: los atributos forman parte de elementos, los textos forman parte de elementos, los elementos a su vez pueden formar parte de otros elementos y así sucesivamente hasta encontrar el elemento raíz del **Document.**

Una vez se ha llegado a este punto, se dispone en memoria de un objeto **Document** que representa la siguiente información presentada con la sintaxis de XML:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<libro disponibleDigital="true" disponiblePapel="true">
<titulo>LENGUAJES DE MARCAS Y ...</titulo>
</libro>
```

5.2. Sistemas de almacenamiento de documentos XML

El manejo de los datos contenidos por un documento XML no siempre se realiza en el momento en el que dichos datos son creados. En muchas ocasiones se desea guardar la información en un sistema de almacenamiento persistente para poder recuperarlo y procesarlo posteriormente.

Las tecnologías que permiten almacenar documentos XML van desde simples ficheros de texto hasta bases de datos nativas XML, pasando por las bases de datos relacionales tradicionales. Todas son válidas, ya que todas consiguen hacer persistente la información para su posterior uso. Como se puede esperar, la forma en la que se logra esta persisten-

© Ediciones Paraninfo

cia es distinta, teniendo cada una de estas alternativas sus ventajas y sus inconvenientes. El acceso a los datos almacenados en un fichero de texto plano debe ser necesariamente distinto al que se realiza a través de una base de datos diseñada de forma específica para trabajar con documentos XML.

El proceso **creación-almacenamiento-recuperación-procesado** de un documento XML se puede, por lo tanto, realizar mediante varias alternativas y habrá que elegir entre ellas. Los parámetros para dicha decisión son tres: la dificultad, considerando la cantidad de esfuerzo que hay que desarrollar y los conocimientos que hay que adquirir para poder usar la solución (en referencia al almacenamiento y lectura, no al procesamiento); la independencia de otras tecnologías, referente a si la solución obliga a tener determinada tecnología disponible en todo momento, normalmente un sistema software, y la capacidad de admitir operaciones de consulta directas realizadas con los lenguajes de consulta específicos para XML.

En la Tabla 5.2 se comparan algunas de las características de las soluciones de almacenamiento de documentos XML citadas.

Tabla 5.2. Características de las soluciones para el manejo de XML

Solución	Dificultad	Proporciona independencia de otras tecnologías	Permite operaciones directas
Ficheros	Baja	Sí	No
Bases de datos	Media	Opcionalmente	Opcionalmente
Bases de datos XML	Media	No	Sí

En los siguientes apartados se muestran las técnicas de recuperación y almacenamiento de documentos XML.

Nota técnica

En esta unidad se utiliza el lenguaje de programación Java para ilustrar algunos ejemplos. Debido a la naturaleza de los documentos XML, se podría utilizar cualquier otro lenguaje que dispusiese de procesadores XML.

5.2.1. Recuperación de documentos XML desde fichero

El proceso completo de **creación-almacenamiento-recuperación-procesado** de documentos XML en fichero se puede resumir gráficamente en el diagrama de la Figura 5.3.

La razón por la que almacenar en un fichero un documento XML es recuperarlo y procesarlo después. Obviamente, leer los caracteres de un fichero y cargarlos en la memoria sin más no supone ninguna ventaja, ya que daría igual utilizar XML que cualquier otro formato. Lo que realmente hace interesante el uso de XML es que hay herramientas que facilitan el trabajo. A la hora de procesar un documento XML desde un programa informático las

© Ediciones Paraninfo

herramientas que se utilizan son los analizadores y existen dos principales: **DOM** y **SAX.** Estos analizadores son independientes del lenguaje, existiendo implementaciones en los lenguajes de programación más habituales.

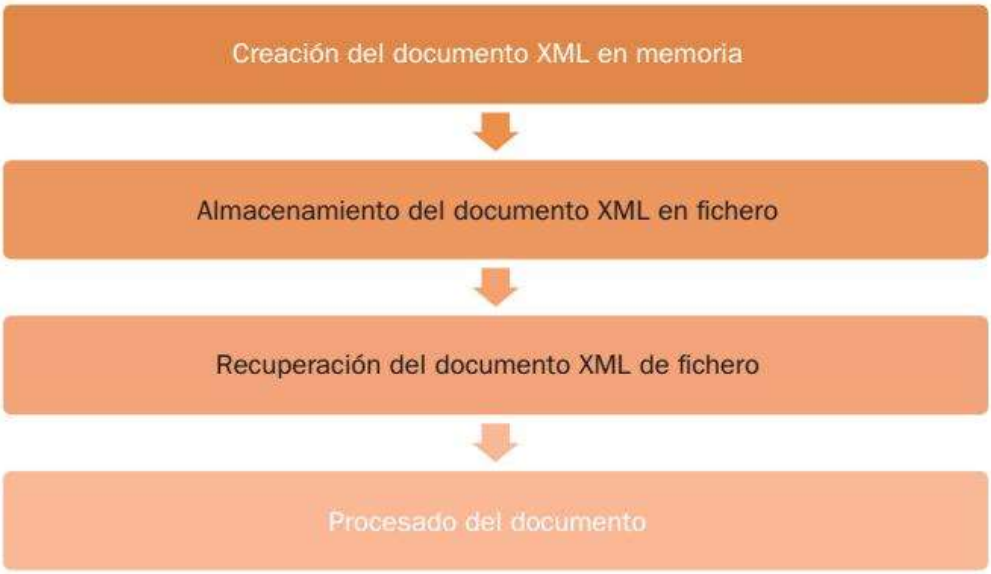


Figura 5.3. *El procesado de un documento XML se suele hacer una vez leído en memoria.*

Estos dos analizadores son distintos en lo referente a cómo realizan el proceso de análisis: leer todo el documento y almacenarlo en memoria para procesarlo posteriormente (analizador DOM) o procesarlo según se realiza la lectura (analizador SAX). El analizador DOM es más versátil y el analizador SAX es más eficiente en términos de uso de memoria. La decisión vendrá determinada por el tipo de proceso que se vaya a realizar o por el tamaño del documento.

Analizador DOM

El analizador DOM necesita cargar el documento XML en memoria en primer lugar para poder procesarlo posteriormente. El siguiente código Java realiza este proceso de carga:

```
File fXmlFile = new File(RUTA_LECTURA + nombreFichero);
DocumentBuilderFactory dbFactory =
DocumentBuilderFactory.newInstance();
DocumentBuilder dBuilder = dbFactory.newDocumentBuilder();
Document document = dBuilder.parse(fXmlFile);
```

El analizador DOM proporciona una colección de métodos para acceder al contenido del documento: elementos, listas de elementos, atributos de elementos o textos. En Java, estos métodos son propios de la interfaz **org.w3c.dom.Document** que hereda, a su vez, de la interfaz **Node**, por lo que también posee sus métodos.

Un objeto de tipo **Document** contiene la representación de un documento XML, proporcionando métodos para acceder a su contenido. Es decir, cuando se carga un documento

© Ediciones Paraninfo

XML en un objeto **Document** se está habilitando un mecanismo para «navegar» por los datos que incluye dicho documento y que están representados en una estructura de datos en forma de árbol.

Los métodos específicos más importantes de la interfaz **Document** para procesar un documento XML son los siguientes:

- getElementById. Proporciona un elemento a partir de su atributo id.
- getElementsByTagName. Proporciona todos los elementos de un determinado tipo.
- getDocumentElement. Proporciona el nodo raíz del documento.

El primer paso del análisis de un documento XML utilizando el analizador DOM consiste en obtener el nodo raíz, que será el punto desde el que se podrá acceder al resto de los nodos del árbol.

```
Element nodoRaiz = document.getDocumentElement();
```

Una vez se tiene acceso al primer elemento del documento, los métodos de acceso a la información serán los propios de la interfaz **Element** y de la interfaz **Node**, de la cual hereda.

Los métodos más relevantes de **Element** para acceder a los datos de un documento son los siguientes:

- getAttribute. Proporciona el valor de un atributo determinado.
- getAttributeNode. Proporciona un atributo (objeto **Attr**) determinado.
- getElementsByTagName. Proporciona todos los elementos que tienen un determinado nombre.
- getTagName. Proporciona el nombre del elemento.
- hasAttribute. Indica si el elemento tiene atributos.

Los métodos más relevantes de **Node** para acceder a los datos de un documento son:

- getAttributes. Proporciona la lista de atributos del nodo.
- getChildNodes. Proporciona la lista de nodos hijos del nodo.
- getFirstChild. Proporciona el primer hijo del nodo.
- getLastChild. Proporciona el último hijo del nodo.
- getNextSibling. Proporciona el nodo siguiente dentro de la rama (nodo «hermano»).
- getNodeName. Proporciona el nombre del nodo.
- getNodeType. Proporciona el tipo de nodo.
- getParentNode. Proporciona el nodo padre del nodo.
- getTextContent. Proporciona el texto contenido en el nodo.
- hasAttributes. Indica si el nodo tiene atributos.
- hasChildNodes. Indica si el nodo tiene nodos hijos.

© Ediciones Paraninfo

Nota técnica

En la **interfaz Node** se declaran las siguientes constantes que determinan el tipo de nodo proporcionado por el método getNodeType.

```
public static final short ELEMENT_NODE = 1;
public static final short ATTRIBUTE_NODE = 2;
public static final short TEXT_NODE = 3;
public static final short CDATA_SECTION_NODE = 4;
public static final short ENTITY_REFERENCE_NODE = 5;
public static final short ENTITY_NODE = 6;
public static final short PROCESSING_INSTRUCTION_NODE = 7;
public static final short COMMENT_NODE = 8;
public static final short DOCUMENT_NODE = 9;
public static final short DOCUMENT_TYPE_NODE = 10;
public static final short DOCUMENT_FRAGMENT_NODE = 11;
public static final short NOTATION_NODE = 12;
```

Actividad resuelta 5.1

Análisis de un documento XML con Java utilizando el analizador DOM

Se tomará como fichero de entrada el siguiente documento XML. La estructura del documento está formada por un elemento raíz (**libros**), una relación de libros (**elementos libro**) con información de disponibilidad en soporte digital o impreso en papel proporcionada por dos atributos (**digital** y **papel**) y un elemento hijo de cada libro (**titulo**) que contiene el título del libro como texto:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<libros editorial="paraninfo">
    <libro id="1" digital="false" papel="true">
        <titulo>LENGUAJES DE MARCAS Y SISTEMAS DE GESTIÓN DE LA INFORMACIÓN</
titulo>
    </libro>
    <!-- ESTO ES UN COMENTARIO -->
    <libro id="2" digital="true" papel="true">
        <titulo>SISTEMAS INFORMÁTICOS</titulo>
    </libro>
    <libro id="3" digital="false" papel="true">
        <titulo>BASES DE DATOS</titulo>
    </libro>
    <libro id="4" digital="true" papel="true">
        <titulo>ENTORNOS DE DESARROLLO</titulo>
    </libro>
    <libro id="5" digital="false" papel="true">
        <titulo>PROGRAMACIÓN</titulo>
    </libro>
    <libro id="6" digital="true" papel="true">
        <titulo>FORMACIÓN Y ORIENTACIÓN LABORAL</titulo>
    </libro>
</libros>
```

El objetivo del ejercicio consiste en mostrar por la consola los títulos de los libros que tienen versión digital.

© Ediciones Paraninfo

Solución

El código que se puede observar a continuación resuelve este problema:

```
Element nodoRaiz = document.getDocumentElement();
NodeList listaLibros = nodoRaiz.getChildNodes();
for (int i = 0; i < listaLibros.getLength(); i++) {
    Node libroActual = listaLibros.item(i);
    if (libroActual.getNodeType() == Node.ELEMENT_NODE) {
        Element elementLibroActual = (Element) libroActual;
        boolean hayDigital =
                Boolean.parseBoolean(elementLibroActual.
                getAttribute("digital"));
        if (hayDigital) {
                System.out.println(elementLibroActual.
                getElementsByTagName("titulo").
                item(0).
                getTextContent());
        }
    }
}
```

A partir del Element denominado **nodoRaiz**, se obtiene una lista con todos los hijos. De estos hijos, en el XML de ejemplo no todos son nodos del tipo **ELEMENT_NODE**: el método **getChildNodes** devuelve todos los hijos, incluidos textos o comentarios, por lo que es necesario filtrar la lista para obtener solo los elementos deseados. Una vez convertidos los nodos adecuados a **Element**, se obtiene el valor del atributo que indica, en el ejemplo, si está disponible en formato digital. Si es así, se accede a los elementos con nombre de etiqueta **titulo** que son hijos de cada elemento (solo habrá uno, por lo que directamente se accederá a la posición 0 del array) y, para finalizar, al contenido textual (el título del libro).

Una solución alternativa para presentar el mismo resultado que en el ejemplo anterior es la siguiente:

```
private void procesarDocumentAlternativo(Document document) {
    Element nodoRaiz = document.getDocumentElement();
    NodeList listaLibros =
        nodoRaiz.getElementsByTagName("libro");
    for (int i = 0; i < listaLibros.getLength(); i++) {
        Element elementLibroActual =
                (Element) listaLibros.item(i);
        boolean hayDigital =
                Boolean.parseBoolean(elementLibroActual.
                getAttribute("digital"));
        if (hayDigital) {
            System.out.println(elementLibroActual.
                    getElementsByTagName("titulo").
                    item(0).getTextContent());
        }
    }
}
```

Como se puede observar, a partir de **nodoRaiz** se obtienen todos los elementos con nombre **libro**, lo que evita tener que comprobar posteriormente si los nodos son de tipo **Element** o no. Esta solución puede presentar un inconveniente: el método **getElementsByTagName** proporciona todos los elementos con nombre **libro** independientemente de la rama del árbol DOM en el que se encuentren. Por lo tanto, si hubiese un elemento **libro** en un nivel más profundo del árbol podría proporcionar un resultado erróneo, ya que en este caso solo se deseaba acceder a los libros del primer nivel de profundidad.

© Ediciones Paraninfo

Recuerda

El método **getElementsByTagName** no siempre resulta adecuado, ya que devuelve todos los elementos con el nombre indicado independientemente de su nivel de profundidad dentro del árbol DOM.

Analizador SAX

El analizador SAX se basa en la lectura secuencial del documento y la invocación automática a los métodos de una interfaz (métodos *callback*). Estos métodos se encuentran definidos en la interfaz **ContentHandler** y se programan heredando de la clase **DefaultHandler.**

El analizador SAX recorre el documento y cuando ocurre un determinado evento llama al método correspondiente. Por ejemplo, al comenzar a analizar el documento se ejecuta **startDocument,** pudiendo programar en este método lo que se desee que ocurra.

Los principales métodos de **DefaultHandler** que hay que sobrescribir se muestran en la Tabla 5.3.

Tabla 5.3. Métodos principales del analizador SAX

Método	Momento de la llamada
startDocument	Al comenzar el análisis del documento.
startElement	Al comenzar el análisis de cada elemento.
characters	Notificación de datos de caracteres dentro de un elemento.
endElement	Al finalizar el análisis de cada elemento.
endDocument	Al finalizar el análisis del documento.

Para analizar un documento con SAX, hay que obtener una instancia del analizador e invocar al método **parse,** indicándole qué fichero se quiere analizar y la instancia de **DefaultHandler** que contiene la implementación de los métodos de análisis:

```
SAXParserFactory saxParserFactory =
     SAXParserFactory.newInstance();
SAXParser saxParser = saxParserFactory.newSAXParser();
DefaultHandler handler = new XMLSAXManager();
File file = new File(nombreFichero);
saxParser.parse(file, handler);
```

En el siguiente ejemplo se muestra el código de una clase heredada de **DefaultHandler** en la que se sobrescriben los métodos principales, algunos de los cuales únicamente muestran un texto que refleja el paso de la ejecución por este.

© Ediciones Paraninfo

```
public class XMLSAXManager extends DefaultHandler {
    private StringBuffer sb;

    @Override
    public void startDocument() throws SAXException {
        System.out.println("Inicio del documento");
    }

    @Override
    public void startElement(String uri, String localName,
        String qName, Attributes atts) throws SAXException {
        if (qName.equals("titulo")) {
            sb = new StringBuffer();
        }
    }

    @Override
    public void characters(char[] ch, int start,
        int length) throws SAXException {
        if (sb != null) {
            sb.append(Arrays.copyOfRange(
                ch, start, start + length));
        }
    }

    @Override
    public void endElement(String uri, String localName,
        String qName) throws SAXException {
        if (qName.equals("titulo")) {
            System.out.println(sb);
        }
    }

    @Override
    public void endDocument() throws SAXException {
        System.out.println("Final del documento");
    }

}
```

Tomando como entrada el fichero XML utilizado en el ejemplo de lectura y análisis con DOM, este analizador mostrará los títulos de todos los libros contenidos en el documento:

```
Inicio del análisis del documento
LENGUAJES DE MARCAS Y SISTEMAS DE GESTIÓN DE LA INFORMACIÓN
SISTEMAS INFORMÁTICOS
BASES DE DATOS
ENTORNOS DE DESARROLLO
PROGRAMACIÓN
FORMACIÓN Y ORIENTACIÓN LABORAL
Final del análisis del documento
```

© Ediciones Paraninfo

Como se puede observar, el análisis es mucho menos versátil que utilizando DOM, pero el hecho de no necesitar tener disponible el documento en memoria de forma constante puede suponer un considerable ahorro de recursos. Por otra parte, el análisis realizado durante la lectura agiliza el proceso, reduciendo los tiempos de ejecución.

Recuerda

DOM y SAX son dos de las API disponibles para analizar documentos XML. Tienen diferentes maneras de realizar dicho análisis y, por lo tanto, se deberá elegir entre una y otra según el tipo de proceso que se quiera realizar. Las ventajas de uno u otro analizador pueden convertirse en desventajas dependiendo del análisis que se vaya a llevar a cabo.

5.2.2. Almacenamiento de documentos XML en fichero

Es el mecanismo más sencillo e independiente, pero a la vez es el que menos posibilidades proporciona a la hora de acceder de manera directa a la información contenida en los documentos XML, ya que cualquier manipulación ha de hacerse con la información cargada en la memoria del programa que lo procesa.

Los documentos XML son, en esencia, una larga cadena de caracteres. Prácticamente todos los lenguajes de programación permiten guardar y recuperar cadenas de caracteres en y desde ficheros de forma relativamente sencilla. Por lo tanto, se puede afirmar que cualquier lenguaje de programación con capacidad para gestionar ficheros puede trabajar con documentos XML. La dificultad aparecerá al realizar el procesamiento de la información, aunque este hándicap se resuelve utilizando procesadores de XML, disponibles prácticamente en todos los lenguajes de programación modernos.

Partiendo de un documento XML creado en memoria, el siguiente paso del proceso **creación-almacenamiento-recuperación-procesado** consiste en almacenarlo en un fichero.

El método escrito en lenguaje Java que se muestra a continuación almacena un objeto **Document** (la representación en memoria de un documento XML) en el fichero **output.xml**, dentro de la ruta especificada en la constante **RUTA_SALIDA.**

```
private void guardarDocument(Document document)
    throws IOException, TransformerException {
        DOMSource domSource = new DOMSource(document);
        FileWriter fileWriter = new FileWriter(
            new File(RUTA_SALIDA + "output.xml"));
        StreamResult streamResult = new
            StreamResult(fileWriter);
        TransformerFactory transformerFactory =
            TransformerFactory.newInstance();
        Transformer transformer =
            transformerFactory.newTransformer();
        transformer.setOutputProperty(OutputKeys.INDENT, "yes");
        transformer.transform(domSource, streamResult);
        fileWriter.close();
}
```


© Ediciones Paraninfo

Las clases específicas del procesamiento de XML tienen los siguientes objetivos: la clase **DOMSource** permite crear un objeto DOM en memoria a partir del árbol creado en el objeto **Document;** la clase **StreamResult** adapta el objeto de la clase **FileWriter** para que almacene el documento XML; las clases **TransformerFactory** y **Transformer** sirven para volcar el objeto de la clase **DOMSource** en el fichero indicado. Por último, es apropiado indicar que la llamada al método **setOutputProperty** del objeto de la clase **Transformer** consigue que el documento XML aparezca sangrado y no en una única línea, lo que dificultaría la lectura por parte de una persona.

Las clases utilizadas en todo este proceso de creación y almacenamiento (además de las clases propias referentes al almacenamiento de ficheros) son las siguientes:

```
import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;
import javax.xml.parsers.ParserConfigurationException;

import org.w3c.dom.Document;
import org.w3c.dom.Attr;
import org.w3c.dom.Element;
import org.w3c.dom.Text;

import javax.xml.transform.dom.DOMSource;
import javax.xml.transform.stream.StreamResult;
import javax.xml.transform.TransformerFactory;
import javax.xml.transform.TransformerException;
import javax.xml.transform.Transformer;
import javax.xml.transform.OutputKeys;
```

5.2.3. Almacenamiento de documentos XML en base de datos convencional

Los sistemas de gestión de base de datos convencionales (relacionales), como Oracle Database, Microsoft SQL Server, MySQL o PostgreSQL, están diseñados para almacenar y gestionar datos a través de tablas y sus relaciones. Son herramientas fundamentales en el funcionamiento de multitud de aplicaciones, ya que permiten estructurar y organizar la información de una manera eficiente y fiable.

El diseño de estos sistemas tiene como objetivo principal el almacenamiento de datos organizados en forma de tablas, divididas a su vez en filas y columnas. De inicio, no están concebidos para alojar documentos estructurados como XML y mucho menos para facilitar el acceso a la información contenida en estos. No obstante, disponen de mecanismos suficientes para almacenar y permitir la recuperación de documentos de cualquier tipo, incluidos los documentos XML, ya que al estar construidos con texto plano el sistema gestor de base de datos más humilde podrá gestionar su almacenamiento sin dificultad. En ocasiones, además, los sistemas de gestión de bases de datos relacionales más avanzados incorporan características específicas para el manejo XML.

© Ediciones Paraninfo

Una solución, por lo tanto, disponible en todos los sistemas de gestión de bases de datos, consiste en almacenar el documento XML en una columna de tipo texto. De esta manera se dispone de una persistencia básica, similar a la proporcionada por los ficheros, pero con las facilidades que proporcionan las bases de datos para manejar grandes volúmenes de información.

Para ilustrar esta solución, se toma como ejemplo un objeto **Document** creado en memoria que contenga el siguiente contenido XML:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<libro disponibleDigital="true" disponiblePapel="true">
<titulo>LENGUAJES DE MARCAS Y ...</titulo>
</libro>
```

El almacenamiento en un campo de tipo texto en una base de datos se podría realizar con el código que se muestra a continuación: un primer bloque de código transforma el objeto **Document** en un **String** que permitirá su almacenamiento en la base de datos; el siguiente bloque de código almacena en una tabla el fichero XML como una cadena de caracteres.

```
// Obtención del String
TransformerFactory transformerFactory = TransformerFactory.newInstance();
Transformer transformer = transformerFactory.newTransformer();
StringWriter stringWriter = new StringWriter();
DOMSource domSource = new DOMSource(document);
transformer.transform(domSource, new StreamResult(stringWriter));
String stringXML = stringWriter.toString();

// Almacenamiento en la base de datos
abrirConexion();
String sql = "INSERT INTO pedidos(nombre, xml) VALUES (?, ?)";
PreparedStatement ps = conexion.prepareStatement(sql);
ps.setString(1, "Librería La Cautiva");
ps.setString(2, stringXML);
ps.executeUpdate();
cerrarConexion();
```

En la Figura 5.4 se muestra el resultado con una captura realizada desde la herramienta de administración de bases de datos MySQL phpMyAdmin:

←T→			id	nombre	xml
☐ Editar	Copiar	Borrar	1	Librería La Cautiva	<?xml version="1.0" encoding="UTF-8" standalone="n...

Figura 5.4. *Al tratarse de texto plano, un documento XML se puede almacenar en cualquier base de datos.*

El proceso de recuperación de la información de la base de datos sería el inverso al de guardado: en primer lugar, hay que realizar la lectura del campo correspondiente; en segundo lugar, hay que convertir el **String** obtenido a objeto **Document**:

© Ediciones Paraninfo

```
// Lectura de la base de datos
abrirConexion();
String sql = "SELECT * from pedidos where id = ?";
PreparedStatement ps = conexion.prepareStatement(sql);
ps.setInt(1, id);
ResultSet resultSet = ps.executeQuery();
resultSet.next();
String stringXML = resultSet.getString("xml");
cerrarConexion();

// Conversión del String leído a objeto Document
DocumentBuilderFactory dbFactory =
     DocumentBuilderFactory.newInstance();
DocumentBuilder dBuilder = dbFactory.newDocumentBuilder();
Document document = dBuilder.parse(
     new InputSource(new StringReader(stringXML)));
```

Como se puede observar, el proceso de almacenamiento en una base de datos convencional difiere del almacenamiento en fichero en la programación de las acciones y en el lugar de almacenamiento de la información, pero no en las facilidades que proporcionan para operar con los datos contenidos en el documento.

Nota técnica

Los documentos XML también pueden almacenarse en base de datos como ficheros utilizando campos de tipo BLOB *(Binary Large Object)* o similares. Estos campos permiten almacenar directamente ficheros de cualquier tipo, incluidos los ficheros XML.

Además, algunos sistemas de bases de datos relacionales como Oracle Database o MySQL permiten realizar consultas XPath sobre documentos XML almacenados en sus tablas, lo que es una manera de integrar XML de manera «nativa» en un sistema gestor de bases de datos no creado específicamente para este fin.

5.2.4. Almacenamiento de documentos XML en base de datos nativa XML

Las bases de datos nativas XML están orientadas al almacenamiento de documentos XML frente a las bases de datos convencionales (relacionales), que están orientadas al almacenamiento de datos organizados en tablas y sus relaciones.

La aportación de las bases de datos nativas XML es la posibilidad de acceder a la información contenida sin necesidad de leer todo el documento, utilizando técnicas de acceso —como por ejemplo el lenguaje **XPath**— que ofrecen una funcionalidad análoga a la proporcionada por el lenguaje SQL en las bases de datos relacionales.

Se dice de las bases de datos nativas XML que están orientadas a documentos frente a las bases de datos relacionales, que están orientadas a datos.

© Ediciones Paraninfo

Existen varios sistemas de gestión de bases de datos XML en el mercado, con licencias tanto libres como comerciales. Algunos de estos sistemas se muestran en la Tabla 5.4.

Tabla 5.4. Algunos sistemas de bases de datos nativas XML

Nombre	Tipo de Licencia
BaseX	Software libre - BSD
eXist-db	Software libre - LGPL
MarkLogic	Comercial
Qualcomm Qizx	Comercial

BaseX es uno de los sistemas gestores de bases de datos nativos XML más reconocidos. Permite crear bases de datos con documentos XML, consultarlos desde diferentes perspectivas, realizar consultas y, por supuesto, editar la información contenida en dichos documentos.

El proceso de instalación de BaseX es muy sencillo. En primer lugar, se debe descargar la aplicación del sitio web: https://basex.org/. Se deberá acceder al portal y pulsar sobre el enlace DOWNLOAD.

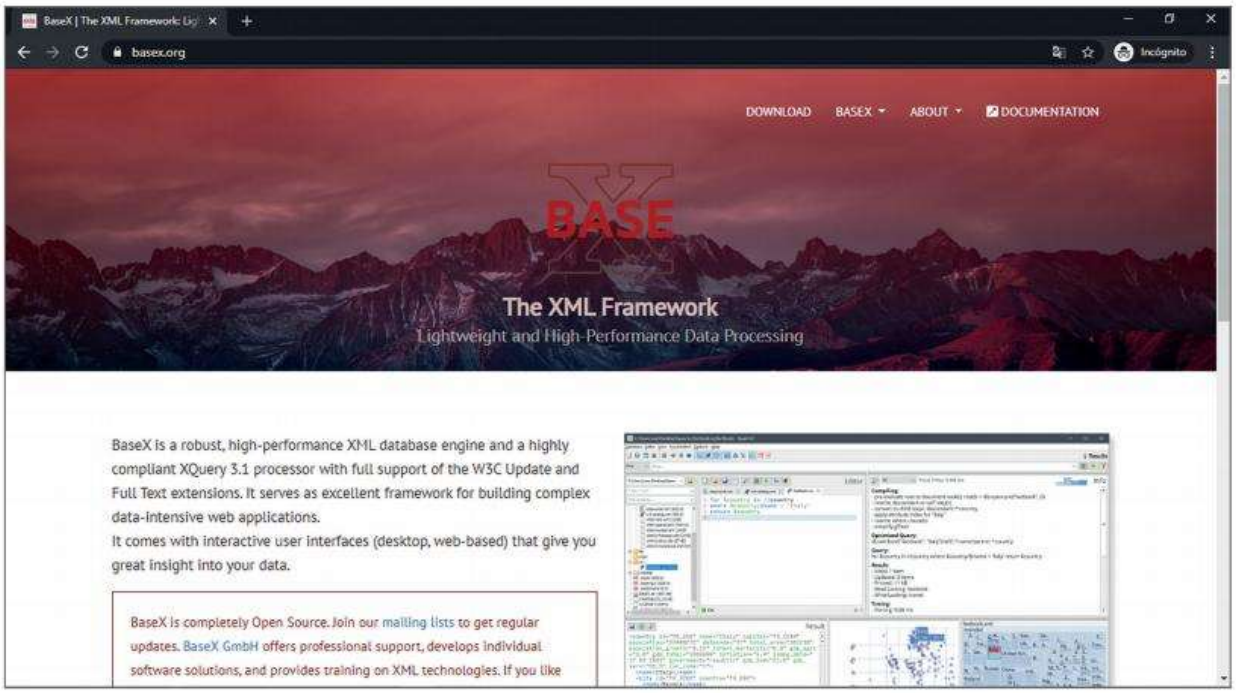


Figura 5.5. *BaseX es una base de datos nativa XML.*

Para realizar la instalación en el sistema operativo Windows, la opción más sencilla es la denominada «Windows Installer», que incluye un instalador automático. Aceptar la configuración propuesta por el asistente será suficiente para instalar BaseX.

© Ediciones Paraninfo

Figura 5.6. *BaseX tiene un proceso de instalación muy sencillo.*

Una vez realizada la instalación, se debe crear la base de datos. Tal y como se muestra en la Figura 5.7, deberá seleccionarse la opción «New» del menú «Database».

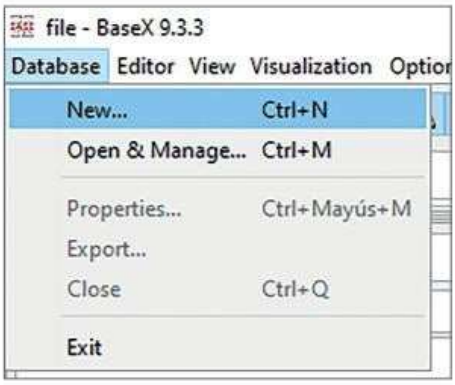


Figura 5.7. *BaseX permite crear y gestionar bases de datos XML.*

A continuación, se deberá seleccionar el fichero XML que se desea procesar. Pulsando sobre el botón «Browse» se podrá seleccionar la ubicación y el fichero XML correspondiente.

En el ejemplo utilizado para mostrar este proceso, el fichero se encuentra ubicado en la ruta D:/BaseX/datos/ y tiene como nombre obras.xml.

© Ediciones Paraninfo

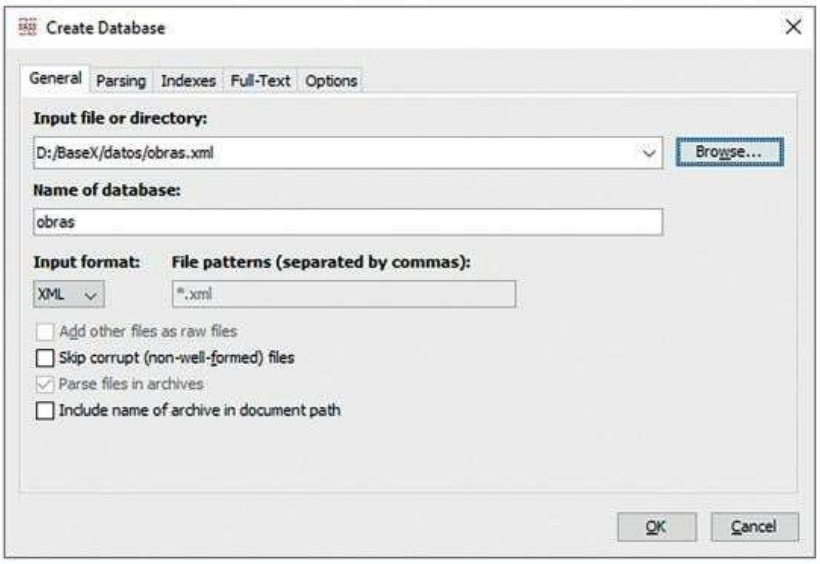


Figura 5.8. *Durante la creación de la base de datos se debe indicar el fichero XML que la almacenará.*

Pulsando el botón «OK», y tras un breve proceso de análisis, BaseX muestra en la interfaz gráfica el documento XML abierto. En este ejemplo, el documento está formado por un conjunto de obras de arte con información sobre el autor, museo en el que se expone, país de origen y, por supuesto, título.

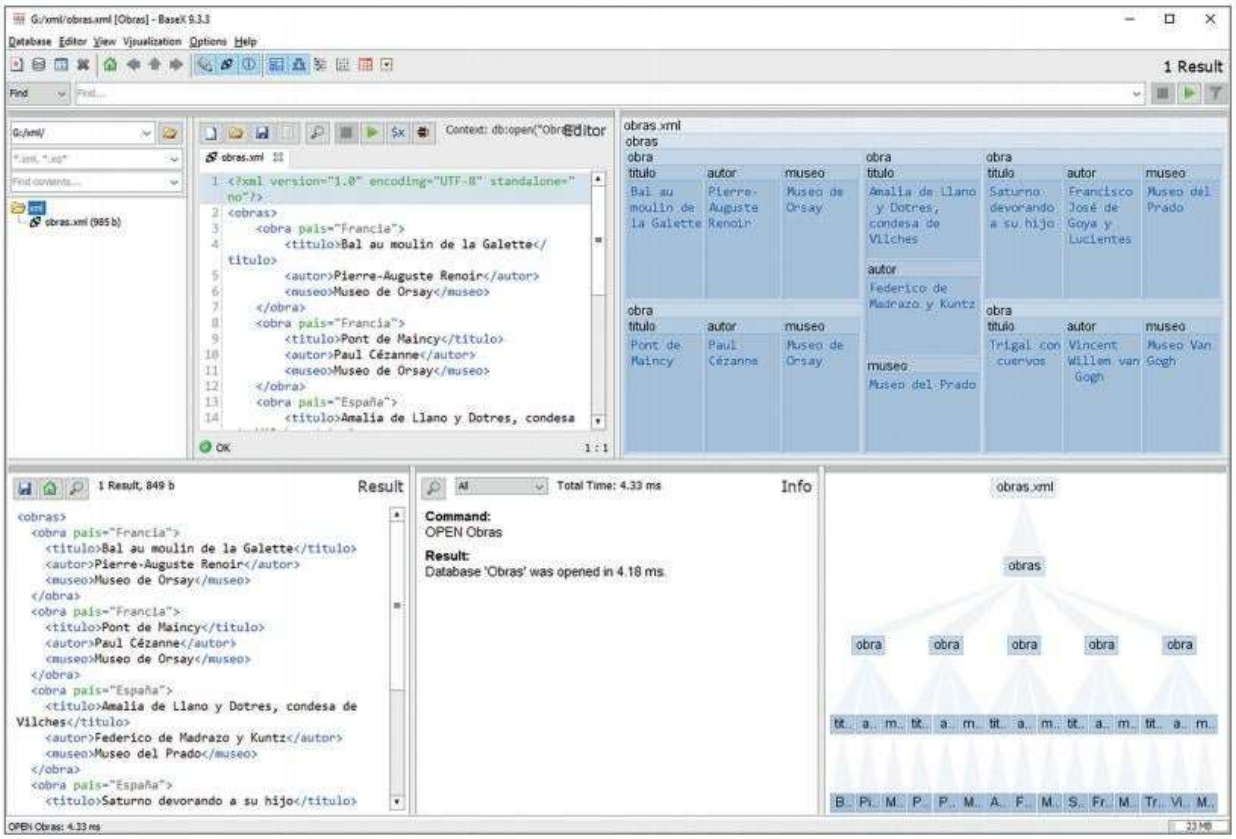


Figura 5.9. *BaseX proporciona diversas perspectivas del mismo documento XML.*

Las vistas proporcionadas son múltiples. De izquierda a derecha y de arriba abajo, los paneles de la configuración inicial de BaseX muestran la siguiente información:

© Ediciones Paraninfo

- Carpeta de trabajo con el listado de documentos XML.
- Documento XML sin procesar (texto).
- Documento XML mostrado como bloques contenedores de elementos que, a su vez, pueden ser contenedores.
- Segmento del documento XML correspondiente a la selección realizada en cualquiera de sus vistas.
- Comando ejecutado y resultado.
- Documento XML mostrado como árbol.

El número de paneles con diferentes presentaciones es amplio y altamente configurable. En la captura de pantalla de la Figura 5.10 se puede observar una configuración distinta en la que se muestra el mismo documento, pero modificando la configuración de la presentación.

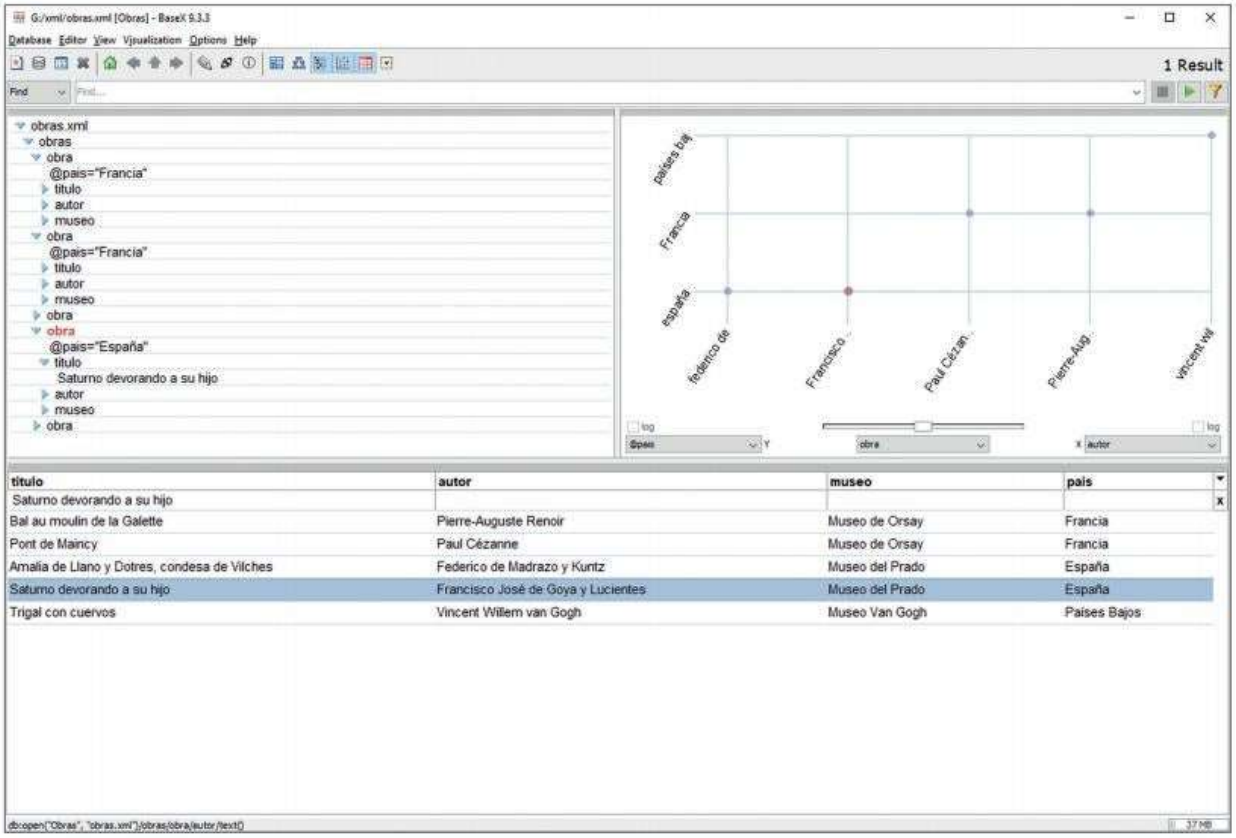


Figura 5.10. *BaseX permite configurar las vistas para adaptarlas a las necesidades del usuario.*

Una de las más interesantes aportaciones de los sistemas de gestión de bases de datos nativas XML es la capacidad de permitir ejecutar consultas utilizando el lenguaje **XQuery.**

Nota técnica

XQuery es un lenguaje de consulta propio de XML, similar a SQL en las bases de datos relacionales.

© Ediciones Paraninfo

En el siguiente ejemplo, se muestra la ejecución de una consulta XQuery en BaseX. En dicha consulta, se desean obtener todas las obras cuyo país de origen sea España. La consulta XQuery de ejemplo es la siguiente: **/obras/obra[@pais='España']**. El resultado proporcionado son todos los elementos **obra** que son hijos de **obras** (nodo principal) y cuyo atributo **pais** tiene como valor **España.**

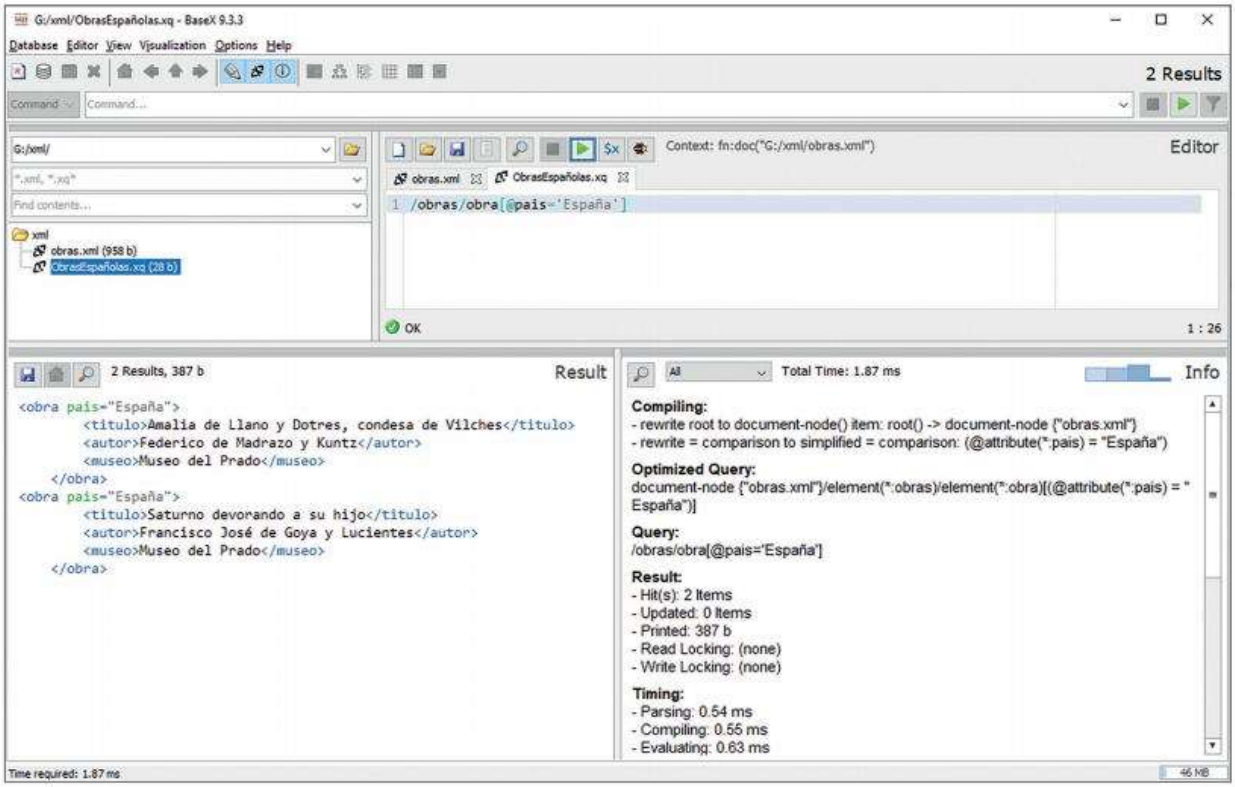


Figura 5.11. *BaseX puede ejecutar consultas con XQuery.*

En el panel inferior izquierdo de la captura de pantalla de la Figura 5.11, se puede observar el resultado de la consulta: un subconjunto del documento XML con aquellos elementos que cumplen con los criterios de búsqueda indicados en la expresión XQuery.

Nota técnica

Debido al auge en la utilización del formato JSON para representar información jerárquica tal y como permite XML, aunque con mucha menos riqueza, algunos de los sistemas de gestión de bases de datos nativos XML han ido evolucionando hacia sistemas capaces de manejar ambos tipos de documentos.

Actividad propuesta 5.1

Creación de una base de datos con BaseX

Esta actividad tiene como objetivo establecer un primer contacto con el sistema gestor de bases de datos BaseX, creando una base de datos en XML.

© Ediciones Paraninfo

- Crea con tu editor de XML favorito un documento con la siguiente estructura:
  - Un elemento principal que contenga un conjunto de series.
  - Por cada serie, el título, el número de temporadas y el año de emisión de la primera temporada.
- El documento debe contener al menos cinco series (incluye *Los Soprano* y *Los Simpson*).
- Instala BaseX en un ordenador.
- Crea una base de datos con el fichero de series.

5.3. Técnicas de búsqueda de información en documentos XML: lenguajes de consulta y manipulación

El lenguaje XML se concibió como un lenguaje para el intercambio de información en un formato estandarizado. Esto es posible debido a que la estructura de los documentos XML es conocida y cumple ciertas reglas, como por ejemplo que la estructura es jerárquica o que los elementos tienen que estar contenidos de manera completa unos dentro de otros (no pueden estarlo parcialmente).

Una de las operaciones más complejas de realizar sobre un documento XML es la búsqueda de información. Mediante el uso de analizadores como SAX o DOM es posible realizar búsquedas de información, pero la forma de hacerlo requiere obligatoriamente conocimientos de programación. Además, suelen ser programas engorrosos de desarrollar y poco eficientes.

Figura 5.12. *Existen sistemas de búsqueda de información en los documentos XML.*

A partir de esta estructura conocida se han desarrollado una serie de tecnologías relacionadas con XML que automatizan tareas como la validación o la transformación. También

© Ediciones Paraninfo

existen tecnologías que automatizan las tareas de búsqueda de información concreta dentro de los documentos XML: XPath y XQuery son dos de ellas. Están muy relacionadas y se complementan. XPath y XQuery proporcionan una herramienta similar a SQL en el acceso a los datos de las bases de datos relacionales.

5.3.1. XPath

XPath es un lenguaje que permite seleccionar nodos o conjuntos de nodos de un documento XML y se utiliza como soporte para otras tecnologías o lenguajes anfitriones como XQuery, XPointer, XSLT o XSL-FO. XPath es una recomendación del W3C y, en sus últimas versiones, permite navegar tanto por documentos XML como por documentos JSON. Cada cierto tiempo el W3C proporciona una nueva versión actualizada de la recomendación. La versión 3.1 fue presentada en marzo de 2017. No obstante, la versión 1.0 de XPath (presentada en 1999) se sigue utilizando de manera habitual en buscadores y herramientas.

Las expresiones XPath no tienen una sintaxis XML, sino una notación similar a la empleada en el establecimiento de rutas en sistemas de archivos o URL. Esto es posible, ya que la estructura de un documento XML es jerárquica de la misma manera que lo es la estructura de un sistema de archivos de un ordenador.

Existen múltiples alternativas para realizar selecciones con XPath: con lenguajes de programación como Java o JavaScript, con herramientas como BaseX o formando parte de transformaciones XSLT, entre otras.

El concepto básico de XPath es la expresión de ruta (*path expression*). Una expresión es una cadena de caracteres construida a partir de palabras clave, símbolos y operandos. XPath, al igual que XML, es sensible a mayúsculas. Las palabras clave de XPath utilizan minúsculas únicamente.

XPath maneja los documentos XML como una estructura en forma de árbol. Dicho árbol está compuesto por nodos y XPath distingue siete tipos de nodos diferentes (esta clasificación corresponde con la versión 3.1 de XPath y difiere ligeramente de versiones anteriores) (Tabla 5.5).

Tabla 5.5. Tipos de nodos de un documento XML

Tipo de nodo	Contenido
Document	Representa el documento XML completo.
Element	Representa los elementos XML.
Attribute	Representa los atributos del documento XML.
Text	Encapsula el contenido textual del documento XML.
Namespace	Representa el enlace de un URI de espacio de nombres a un prefijo de espacio de nombres o al espacio de nombres predeterminado.
Processing instruction	Encapsula instrucciones de procesamiento XML.
Comment	Encapsula comentarios XML.

© Ediciones Paraninfo

Con el objetivo de ilustrar las expresiones XPath se tomará el siguiente documento XML como ejemplo. En él se representan varias obras pictóricas:

```
<?xml version="1.0" encoding="UTF-8"?>
<obras>
   <obra pais="Francia" anyo="1876">
      <titulo>Bal au moulin de la Galette</titulo>
      <autor>Pierre-Auguste Renoir</autor>
      <museo visible="true">Museo de Orsay</museo>
   </obra>
   <obra pais="Francia" anyo="1879">
      <titulo>Pont de Maincy</titulo>
      <autor>Paul Cézanne</autor>
      <museo visible="false">Museo de Orsay</museo>
   </obra>
   <obra pais="España" anyo="1822">
      <titulo>Amalia de Llano y Dotres, condesa de Vilches</titulo>
      <autor>Federico de Madrazo y Kuntz</autor>
      <museo visible="true">Museo del Prado</museo>
   </obra>
   <obra pais="España" anyo="1823">
      <titulo>Saturno devorando a su hijo</titulo>
      <autor>Francisco José de Goya y Lucientes</autor>
      <museo visible="true">Museo del Prado</museo>
   </obra>
   <obra pais="España" anyo="1635">
      <titulo>El príncipe Baltasar Carlos a caballo</titulo>
      <autor>Diego Rodríguez de Silva y Velázquez</autor>
      <museo visible="false">Museo del Prado</museo>
   </obra>
   <obra pais="Países Bajos" anyo="1890">
      <titulo>Trigal con cuervos</titulo>
      <autor>Vincent Willem van Gogh</autor>
      <museo visible="true">Museo Van Gogh</museo>
   </obra>
</obras>
```

El lenguaje XPath es muy extenso y sus expresiones están categorizadas para una mejor localización y comprensión. Las categorías son las siguientes:

- Expresiones de ruta de localización («*location path expressions*»).
- Expresiones literales.
- Expresiones condicionales.
- Expresiones lógicas.
- Expresiones de secuencias.
- Expresiones «for».
- Expresiones cuantificadas.

© Ediciones Paraninfo

- Expresiones de comparación.
- Expresiones aritméticas.
- Expresiones de concatenación de cadenas.
- Expresiones de asignación.
- Expresiones en secuencias de tipos.

Los tipos de funciones que proporciona XPath son los siguientes:

- Tipo de cadenas de caracteres.
- Funciones que manejan URI.
- Funciones que manejan valores booleanos.
- Funciones que manejan periodos de tiempo.
- Funciones que manejan fechas y horas.
- Funciones relativas a QNames *(qualified names)*.
- Funciones relativas a nodos.
- Funciones relativas a secuencias.
- Funciones de contexto.
- Funciones relativas a mapas y arrays.
- Funciones de manejo de datos JSON.

Los tipos de operadores que proporciona XPath son los siguientes:

- Operadores numéricos.
- Operadores de cadenas de caracteres.
- Operadores booleanos.
- Operadores de periodos de tiempo.
- Operadores de fecha y horas.
- Operadores de nodos.
- Operadores de secuencias.

Supone, por lo tanto, una gran inversión de tiempo y esfuerzo llegar a dominar XPath. Dado que está fuera del alcance de este libro ser exhaustivo en el conocimiento de este lenguaje, se van a presentar los conocimientos más útiles y cuyo uso es más habitual. El estándar completo de XPath y su especificación se define en las páginas siguientes:

- Principal: https://www.w3.org/TR/xpath/all/
- Recomendación 3.1: https://www.w3.org/TR/2017/REC-xpath-31-20170321/
- Modelo de datos 3.1: https://www.w3.org/TR/xpath-datamodel-31/

Las expresiones de rutas de localización son, sin ninguna duda, las de uso más común. Permiten recuperar parte de un documento con fórmulas similares a las utilizadas en la expresión de rutas en sistemas de archivos, pero incluyendo una mayor riqueza.

© Ediciones Paraninfo

Las expresiones de ruta están formadas por tres partes:

- Un eje (axis). La dirección de la navegación.
- Una prueba de nodo (node test). El filtro que determina el conjunto de nodos del total de nodos situados en el eje.
- Cero o más predicados (predicates). Mecanismo para hacer un filtro sobre los nodos proporcionados por el eje y la prueba de nodo.

La sintaxis es, por lo tanto, la siguiente:

```
axis::node_test[predicate]
```

Una expresión de ruta puede comenzar opcionalmente por «/» o por «//». Está formada por uno o más pasos separados por «/» o «//».

«/» al comienzo de una expresión indica, al igual que en los sistemas de archivos, el nodo raíz del documento. «//» al comienzo de una expresión permitiría, partiendo del nodo raíz, seleccionar todos los nodos hijos que tengan unas características determinadas.

El funcionamiento de «/» y «//» en medio de la ruta es distinto: mientras que «/» es un separador de ruta, «//» permitiría, al igual que cuando se utiliza en el comienzo de la expresión, obtener todos los nodos hijos que tengan unas características determinadas, pero a partir de la posición actual en lugar de a partir del nodo raíz.

En la Tabla 5.6 se muestran algunas expresiones de ruta sencillas y las descripciones asociadas. Todas las expresiones podrán devolver «vacío» si no hay concordancia entre la expresión y el documento XML.

Tabla 5.6. Expresiones XPath

Expresión	Descripción
elemento	Proporciona el nodo **elemento** a partir de la posición actual.
/elemento	Proporciona el nodo **elemento** indicado a partir de la raíz del documento.
//elemento	Proporciona todos los nodos **elemento** independientemente de su posición.
/elemento1/elemento2	Proporciona todos los nodos **elemento2** que son hijos del elemento raíz **elemento1.**
/elemento1//elemento2	Proporciona todos los nodos **elemento2** que dependen jerárquicamente de **elemento1**, independientemente de su nivel.
//elemento1/../elemento2	Proporciona los nodos **elemento2** que son hijos de los padres de los nodos **elemento1.**

En la Tabla 5.7 se muestran varios ejemplos que ilustran el uso de las expresiones XPath utilizando como entrada el documento XML anterior. El * sirve como comodín, al igual

© Ediciones Paraninfo

que en las rutas de los sistemas de archivos. Estas expresiones contendrían solo rutas simples y no sujetas a condiciones.

Recuerda

El nombre XPath tiene su origen en el mecanismo de direccionamiento jerárquico utilizado, similar a una expresión de una ruta.

Tabla 5.7. Ejemplos de expresiones XPath

Expresión	Resultado
*	Todo el documento.
/	Todo el documento.
/obras	Nodos **obras** y todos sus descendientes.
//obras	Todos los nodos **obras**, independientemente de en qué lugar se encuentren, y todos sus descendientes.
/obra	Vacío. Los nodos **obra** no están en la raíz del documento.
/obras/museo	Vacío. Los nodos **museo** no son hijos directos del nodo **obras** en el documento de ejemplo.
/obras//museo	Todos los nodos museo que dependan jerárquicamente, ya sea de forma directa o indirecta, del nodo **obra.**
/obras/obra/*	Todos los nodos que dependen jerárquicamente de los nodos **obra** que, a su vez, dependen directamente del nodo **obras.**

Se pueden utilizar atributos en las expresiones. Estos van precedidos del símbolo @ y admiten condiciones (Tabla 5.8).

Tabla 5.8. Ejemplos de expresiones XPath que utilizan atributos

Expresión	Resultado
//@*	Todos los atributos de todos los nodos.
//obra/@*	Todos los atributos de todos los nodos **obra.**
//obra/@pais	Atributo **pais** de todos los nodos **obra.**
//obra[@pais='Francia']	Todos los nodos **obra** cuyo atributo **país** tiene como valor **'Francia'.**
//museo[@visible='true']/../titulo/text()	Todos los títulos (solo el texto) de aquellas **obras** que están en **museos** cuyo atributo **visible** tiene como valor true.

El conjunto de funciones denominado «funciones de cadena» es extenso, incluyendo expresiones regulares. Algunas de las funciones más útiles son las que se recogen en la Tabla 5.9.

© Ediciones Paraninfo

Tabla 5.9. Funciones XPath que manejan cadenas de caracteres

Función	Descripción
compare	Devuelve un valor numérico (–1, 0, 1) que determina si dos cadenas son iguales (valor 0) o cuál de las dos iría antes que la otra en función de la ordenación elegida.
substring	Devuelve un subconjunto de caracteres de la cadena.
string-length	Devuelve el número de caracteres de una cadena.
upper-case	Devuelve una cadena de caracteres convertida en mayúsculas.
lower-case	Devuelve una cadena de caracteres convertida en minúsculas.
contains	Indica si una cadena contiene o no determinado texto.
starts-width	Indica si una cadena comienza por un determinado conjunto de caracteres.
end-width	Indica si una cadena termina con un determinado conjunto de caracteres.

Algunos ejemplos de expresiones que utilizan funciones de cadena se muestran en la Tabla 5.10.

Tabla 5.10. Ejemplos de funciones XPath que manejan cadenas de caracteres

Expresión	Resultado
//obra/titulo/string()	Todos los **títulos** (solo el texto) de todas las **obras.**
//obra/@pais/string()	Todos los valores (solo el texto) de los atributos **pais** de los nodos **obra.**
//obra/@pais/text()	Vacío. La función text() no tiene efecto sobre los atributos.
//titulo[starts-with(text(), 'S')]	Todos los títulos de las **obras** que comienzan por **S** mayúscula.
//titulo[contains(text(), 'h')]	Todos los títulos de las **obras** que contienen una **h** minúscula.
//titulo/substring(text(),1,3)	Primeros tres caracteres de cada **título.**
//obra/@pais/upper-case(string())	Todos los valores (solo el texto) convertidos a mayúsculas de los atributos **pais** de los nodos **obra.**
//obra/@pais/string-length(string())	Indica la longitud de los atributos **pais** de todos los nodos **obra.**

Otro conjunto de funciones es el denominado «funciones de contexto». Algunas de las funciones de este grupo son las que se indican en la Tabla 5.11.

Tabla 5.11. Funciones XPath de contexto

Función	Descripción
position	Proporciona la posición de un nodo en su conjunto.
last	El último nodo de un conjunto.
[posicion]	Hace referencia a la posición del nodo dentro de un conjunto.

© Ediciones Paraninfo

Algunos ejemplos de uso de «funciones de contexto» se recogen en la Tabla 5.12.

Tabla 5.12. Ejemplos de funciones XPath de contexto

Expresión	Resultado
//obra[1]	Primera **obra** de todos los nodos **obra.**
//obra[position()=3]	El nodo **obra** que se encuentra en tercera posición.
//obra[last()]	El nodo **obra** que se encuentra en última posición.

Las denominadas «funciones agregadas» toman como entrada una secuencia de elementos y devuelven un único valor. Son las recogidas en la Tabla 5.13.

Tabla 5.13. Funciones agregadas de XPath

Función	Descripción
count	Número de elementos.
avg	Media.
max	Valor máximo.
min	Valor mínimo.
sum	Acumulado.

Algunos ejemplos de uso de «funciones agregadas» se recogen en la Tabla 5.14.

Tabla 5.14. Ejemplos de funciones agregadas de XPath

Expresión	Resultado
sum(//[@anyo])	La suma de todos los valores de los atributos **anyo.**
max(//[@anyo])	El valor más alto de todos los atributos **anyo.**
avg(//[@anyo])	El valor promedio de todos los atributos **anyo.**

XPath admite también operadores. En la Tabla 5.15 se muestra un conjunto representativo de estos.

Tabla 5.15. Operadores de XPath

Operador	Tipo	Descripción
+	Aritmético	Suma.
-	Aritmético	Resta.
*	Aritmético	Multiplicación.

© Ediciones Paraninfo

Operador	Tipo	Descripción
div	Aritmético	División.
idiv	Aritmético	División entera.
mod	Aritmético	Resto de división (módulo).
+	Aritmético	Devuelve el operando con el signo sin cambios (utilizado para conversión de tipos).
-	Aritmético	Devuelve el operando con el signo cambiado.
=	Lógico	Iguales.
!=	Lógico	Distintos.
>	Lógico	Mayor que.
<	Lógico	Menor que.
>=	Lógico	Mayor o igual.
<=	Lógico	Menor o igual.
and	Lógico	Y lógica.
or	Lógico	O lógica.
not	Lógico	Negación.
name	De manejo de nodos	Nombre del nodo.
local-name	De manejo de nodos	Parte local del nombre del nodo.
namespace-uri	De manejo de nodos	URI del espacio de nombres del nombre del nodo.
root	De manejo de nodos	Devuelve la raíz del nodo.
path	De manejo de nodos	Devuelve la expresión de ruta relativa para localizar el nodo seleccionado.
has-children	De manejo de nodos	Indica si el nodo seleccionado tiene hijos.

En la Tabla 5.16 se recogen algunos ejemplos de expresiones que utilizan operadores.

Tabla 5.16. Ejemplos de operadores de XPath

Expresión	Resultado
//@anyo[data()>1850]/..	Todos los nodos obra cuyo atributo anyo tiene un valor superior a 1850.
count(//@anyo[data()>1850]/..)	Número de obras cuyo atributo anyo tiene un valor superior a 1850.
//@anyo[data()>1850 and data()<1880]/..	Obras cuyo atributo anyo tiene como valor un número entre 1850 y 1880.

© Ediciones Paraninfo

Expresión	Resultado
//obra/titulo[text()>"N"]/string()	Los títulos de la obra que empiecen por una letra mayor que "N".
//obra[@pais="Francia"]/(@anyo + 100)	Suma **100** al atributo **anyo** de las **obras** cuyo **pais** es **"Francia".**
//obra[@anyo=1876]	Selecciona los nodos **obra** cuyo atributo **anyo** tiene como valor **1876.**
//obra[@anyo!=1876]	Selecciona los nodos **obra** cuyo atributo **anyo** tiene como valor distinto a **1876.**
//obra[@anyo<=1876]	Selecciona los nodos **obra** cuyo atributo **anyo** tiene como valor un número inferior o igual a **1876.**
//obra[not(@anyo=1876) and not (@anyo=1879)]	Selecciona los nodos **obra** cuyo atributo **anyo** no tiene como valor **1876** ni **1879.**
//obra[(@pais>"España")]	Selecciona los nodos **obra** cuyo atributo **pais** tiene un valor posterior alfabéticamente a **España.**
name(//*[@anyo=1876])	El nombre del nodo cuyo atributo **anyo** tiene **1876** como valor.
has-children(//*[@anyo=1876])	Indica si el nodo cuyo atributo **anyo** con valor **1876** tiene nodos hijo.

Por otra parte, los ejes disponibles se muestran en la siguiente lista. Los ejes permiten «navegar» de manera relativa a partir de una posición en el árbol:

- Ejes hacia delante (*forward axis*):
  - child.
  - descendant.
  - attribute.
  - self.
  - descendant-or-self.
  - following-sibling («hermanos» siguiente).
  - following.
  - namespace.
- Ejes hacia atrás (*reverse axis*):
  - parent.
  - ancestor.
  - preceding-sibling («hermanos» anteriores).
  - preceding.
  - ancestor-or-self.

En la Tabla 5.17 se muestran algunos ejemplos de acceso mediante ejes.

© Ediciones Paraninfo

Tabla 5.17. Ejemplos de acceso mediante ejes

Expresión	Resultado de la evaluación
//obra/child::autor	Todos los nodos **autor** que sean «hijos» de nodos obra.
//obra/child::*	Todos los nodos «hijo» (**titulo, autor** y **museo**) de todos los nodos obra.
//obra/child::autor[starts-with(text(), 'F')]	Todos los nodos **autor** que sean «hijos» de nodos y cuyo nombre comience por la letra «F».
//museo[@visible="true"]/parent::*	Los nodos **obra** (son los nodos «padre» de aquellos nodos **museo** cuyo atributo **visible** tenga como valor **true.**
obras/obra[@anyo=1822]/titulo/following-sibling::*	Los nodos **autor** y **museo** (son los nodos «hermano» siguiente de **título**) de la **obra** cuyo atributo **anyo** sea **1822.**

Como se puede comprobar, la riqueza de XPath es enorme y permite encontrar una expresión para obtener prácticamente cualquier conjunto de elementos de un documento XML en función de las más complejas condiciones.

Nota técnica

XPath en su versión 3.1 es una tecnología muy extensa. En esta unidad se expone un subconjunto de sus funciones, pero hay muchas más, aunque estas exceden del ámbito de este libro.

Actividad resuelta 5.2

Creación de consultas con XPath

Tomando como fichero de entrada el documento XML proporcionado en este enunciado en el que se recogen algunos de los videojuegos más vendidos de la historia, y utilizando BaseX como herramienta para comprobar el buen funcionamiento de las soluciones, crear las siguientes expresiones XPath:

- Mostrar todos los videojuegos.
- Mostrar los títulos de todos los videojuegos.
- Mostrar todos los títulos de videojuego que contentan la letra «T».
- Mostrar todos los títulos de videojuego que se hayan publicado a partir del año 2000.

Documento XML:

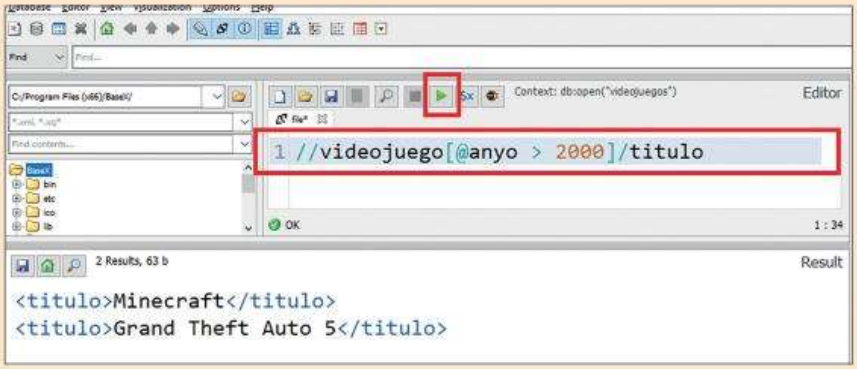
```
<videojuegos>
    <videojuego anyo="2011">
       <titulo>Minecraft</titulo>
        <genero>Sandbox</genero>
```

© Ediciones Paraninfo

```
        <unidades-vendidas>176000000</unidades-vendidas>
    </videojuego>
    <videojuego anyo="1984">
        <titulo>Tetris</titulo>
        <genero>Puzle</genero>
        <unidades-vendidas>170000000</unidades-vendidas>
    </videojuego>
    <videojuego anyo="2013">
        <titulo>Grand Theft Auto 5</titulo>
        <genero>Sandbox</genero>
        <unidades-vendidas>110000000</unidades-vendidas>
    </videojuego>
    <videojuego anyo="1985">
        <titulo>Super Mario Bros</titulo>
        <genero>Plataformas</genero>
        <unidades-vendidas>48000000</unidades-vendidas>
    </videojuego>
</videojuegos>
```

Solución

Una vez creada la base de datos con el archivo XML proporcionado, escribir las expresiones en la ventana «Editor» y pulsar el botón «Run Query» para comprobar que los resultados son los esperados.



- Mostrar todos los videojuegos.

  `//videojuego`

- Mostrar los títulos de todos los videojuegos.

  `//videojuego/titulo`

- Mostrar todos los títulos de videojuego que contentan la letra «T».

  `//videojuego/titulo[contains(text(),'T')]`

- Mostrar todos los títulos de videojuego que se hayan publicado a partir del año 2000.

  `//videojuego[@anyo > 2000]/titulo`

Actividad propuesta 5.2

Creación de consultas con XPath

Esta actividad tiene como objetivo adquirir soltura en la creación de expresiones XPath.

© Ediciones Paraninfo

Tomando como fichero de entrada la solución de la Actividad propuesta 5.1, correspondiente a las series, y utilizando BaseX como herramienta para comprobar el buen funcionamiento de las soluciones, crea las siguientes expresiones XPath.

- Mostrar todas las series.
- Mostrar todas las series que se estrenaron antes de un año determinado.
- Mostrar todas las series que tienen más de diez temporadas.
- Mostrar todas las series cuyo título contenga la letra «L» en cualquier posición.

5.3.2. XQuery

XQuery es un lenguaje de consulta para acceder a la información contenida en un documento XML. Es el equivalente a SQL en el acceso a los datos de las bases de datos relacionales. Dispone de capacidades de programación. XQuery es también un lenguaje de programación funcional, permitiendo la ejecución de instrucciones. XQuery es, además, una recomendación del W3C.

Comparte con XPath el modelo de datos, las funciones y los operadores. XQuery es un superconjunto de XPath, por lo que se puede afirmar que toda expresión XPath es, asimismo, una expresión XQuery.

Utilizando XQuery se puede, por ejemplo, extraer una sección de la información proporcionada por un servicio web, transformar un documento XML en XHTML o buscar información en páginas web.

Algunas de las características más relevantes de la sintaxis de XQuery son las siguientes:

- Las cadenas de caracteres pueden estar rodeadas de comillas simples o dobles.
- Distingue entre mayúscula y minúsculas.
- Las variables comienzan por el símbolo $.

Cada expresión XQuery la evalúa un procesador y se ejecuta sobre un documento XML para proporcionar un conjunto de datos. Dichas expresiones utilizan XPath para acceder a los distintos contenidos del documento, tal y como se hace con el lenguaje SQL sobre los datos almacenados en las tablas de las bases de datos relacionales.

Las cláusulas FLWOR (acrónimo de *For, Let, Where, Order by, Return*) proporcionan a XQuery toda la potencia necesaria para hacer todo tipo de operaciones de consulta sobre documentos XML. Estas cláusulas son tantas como las siglas de la palabra FLWOR:

- for, para recorrer colecciones. Asocia una variable con cada elemento devuelto por la consulta.
- let, para vincular resultados a variables.
- where, para incorporar condiciones a las cláusulas for.
- order by, para ordenar los resultados.
- return, para indicar qué resultados se han de devolver.

© Ediciones Paraninfo

La sintaxis de una cláusula FLWOR es la siguiente:

- Una o más cláusulas **for** o **let** (obligatoria).
- Una cláusula **where** (opcional).
- Una cláusula **order by** (opcional).
- Una cláusula **return** (obligatoria).

Lógicamente, estas cláusulas deben ejecutarse sobre un documento previamente referenciado. La función **doc** permite realizar dicha referencia. Aplicada sobre un documento XML devolverá un **Document** sobre el que se podrá indicar una ruta para acceder a un subconjunto de elementos.

Se utilizará BaseX como herramienta para la ejecución de consultas XQuery, conjuntamente con el documento **obras.xml.**

El siguiente ejemplo, quizá el más simple sintácticamente posible, devolvería todo el documento XML:

```
for $a in doc("obras.xml")
return $a
```

Retorno:

```
<obras>
  <obra pais="Francia" anyo="1876">
    <titulo>Bal au moulin de la Galette</titulo>
    <autor>Pierre-Auguste Renoir</autor>
    <museo visible="true">Museo de Orsay</museo>
  </obra>
  <obra pais="Francia" anyo="1879">
    <titulo>Pont de Maincy</titulo>
    <autor>Paul Cézanne</autor>
    <museo visible="false">Museo de Orsay</museo>
  </obra>
...
```

La consulta siguiente no devolverá el documento completo, sino los nodos **obra** contenidos en este:

```
for $a in doc("obras.xml")//obra
return $a
```

El resultado en este caso sería el siguiente:

```
<obra pais="Francia" anyo="1876">
  <titulo>Bal au moulin de la Galette</titulo>
  <autor>Pierre-Auguste Renoir</autor>
```


© Ediciones Paraninfo

```
  <museo visible="true">Museo de Orsay</museo>
</obra>
<obra pais="Francia" anyo="1879">
  <titulo>Pont de Maincy</titulo>
  <autor>Paul Cézanne</autor>
  <museo visible="false">Museo de Orsay</museo>
</obra>
```

Una cláusula FLWOR más elaborada sería:

```
for $a in doc("obras.xml")//obra
where $a/@anyo>"1800"
order by $a/@pais
return data($a/@pais)
```

Como se puede observar, la cláusula anterior incluye una condición (**where**) y una ordenación (**order by**). En ambos casos, se referencia a la variable **$a** que contiene todos los nodos **obra** del documento. El resultado de esta expresión serían todos los valores de los atributos **pais** de todas las **obras** cuyo atributo **anyo** sea mayor que **1800.** Los datos se muestran ordenados por dicho atributo **pais**, tal y como se puede comprobar en el resultado que se muestra a continuación:

```
España
España
Francia
Francia
Países Bajos
```

Otra posible consulta sería la siguiente:

```
for $a in doc("obras.xml")//obra
where $a/@anyo>"1800"
return data($a)
```

En este caso, el resultado obtenido son los textos contenidos en los nodos **obra** cuyo atributo **anyo** tenga un valor mayor que **1800**:

```
Bal au moulin de la GalettePierre-Auguste RenoirMuseo de Orsay
Pont de MaincyPaul CézanneMuseo de Orsay
Amalia de Llano y Dotres, condesa de VilchesFederico de Madrazo y KuntzMuseo
del Prado
Saturno devorando a su hijoFrancisco José de Goya y LucientesMuseo del Prado
Trigal con cuervosVincent Willem van GoghMuseo Van Gogh
```

Sobre XQuery es posible realizar construcciones más complejas. Por ejemplo, se puede crear documentos HTML a partir de los resultados obtenidos por las consultas. Bastará con

© Ediciones Paraninfo

incrustar los valores que se deseen mostrar entre llaves {} e intercalar código HTML con las expresiones XQuery.

En el siguiente ejemplo, se construye una lista sobre el resultado de la consulta de obras anterior:

```
<ul>
{
  for $a in doc("obras.xml")//obra
  where $a/@anyo>"1800"
  return <li>{data($a)}</li>
}
</ul>
```

El resultado es una sección bien construida de una página HTML en la que se muestra una lista con los datos seleccionados:

```
<ul>
  <li>Bal au moulin de la GalettePierre-Auguste RenoirMuseo de Orsay</li>
  <li>Pont de MaincyPaul CézanneMuseo de Orsay</li>
  <li>Amalia de Llano y Dotres, condesa de VilchesFederico de Madrazo y
KuntzMuseo del Prado</li>
  <li>Saturno devorando a su hijoFrancisco José de Goya y LucientesMuseo del
Prado</li>
  <li>Trigal con cuervosVincent Willem van GoghMuseo Van Gogh</li>
</ul>
```

Otro ejemplo similar consiste en la creación del código HTML de una tabla con los valores de los elementos **titulo** y **autor** ordenados por autor:

```
<table>
{
  for $a in doc("datos/obras.xml")/obras/obra
    order by $a/autor
    return <tr><td>{$a/titulo/text()}</td><td>{$a/autor/text()}</td></tr>
}
</table>
```

El resultado de la ejecución es el siguiente:

```
<table>
  <tr>
    <td>El príncipe Baltasar Carlos a caballo</td>
    <td>Diego Rodríguez de Silva y Velázquez</td>
  </tr>
  <tr>
    <td>Amalia de Llano y Dotres, condesa de Vilches</td>
```


© Ediciones Paraninfo

```
    <td>Federico de Madrazo y Kuntz</td>
  </tr>
  <tr>
    <td>Saturno devorando a su hijo</td>
    <td>Francisco José de Goya y Lucientes</td>
  </tr>
  <tr>
    <td>Pont de Maincy</td>
    <td>Paul Cézanne</td>
  </tr>
  <tr>
    <td>Bal au moulin de la Galette</td>
    <td>Pierre-Auguste Renoir</td>
  </tr>
  <tr>
    <td>Trigal con cuervos</td>
    <td>Vincent Willem van Gogh</td>
  </tr>
</table>
```

XQuery también admite la inserción de sentencias condicionales if-then-else. El siguiente ejemplo es una modificación del anterior, generando solamente las filas correspondientes a las obras cuyo atributo **pais** tiene como valor **Francia.**

```
<table>
{
  for $a in doc("datos/obras.xml")/obras/obra
  order by $a/autor
  return if ($a/@pais="Francia") then
  <tr><td>{$a/titulo/text()}</td><td>{$a/autor/text()}</td></tr>
}
</table>
```

En este caso el resultado proporcionado es el siguiente:

```
<table>
  <tr>
    <td>Pont de Maincy</td>
    <td>Paul Cézanne</td>
  </tr>
  <tr>
    <td>Bal au moulin de la Galette</td>
    <td>Pierre-Auguste Renoir</td>
  </tr>
</table>
```

© Ediciones Paraninfo

No es objetivo de este libro hacer una presentación exhaustiva de XQuery, ya que esto se encuentra fuera de su ámbito. No obstante, se ofrece una presentación suficiente para comprender el potencial y la capacidad de este lenguaje.

5.3.3. XLink y XPointer

XLink y XPointer son otras dos tecnologías relacionadas con XML que permiten navegar por los documentos XML y sus elementos.

XLink sirve para incluir enlaces en documentos XML. XPointer, por su parte, permite «apuntar» a partes determinadas de un documento XML utilizando expresiones XPath.

Empleando conjuntamente XLink y XPointer, se puede crear un enlace a un elemento determinado de un documento de destino.

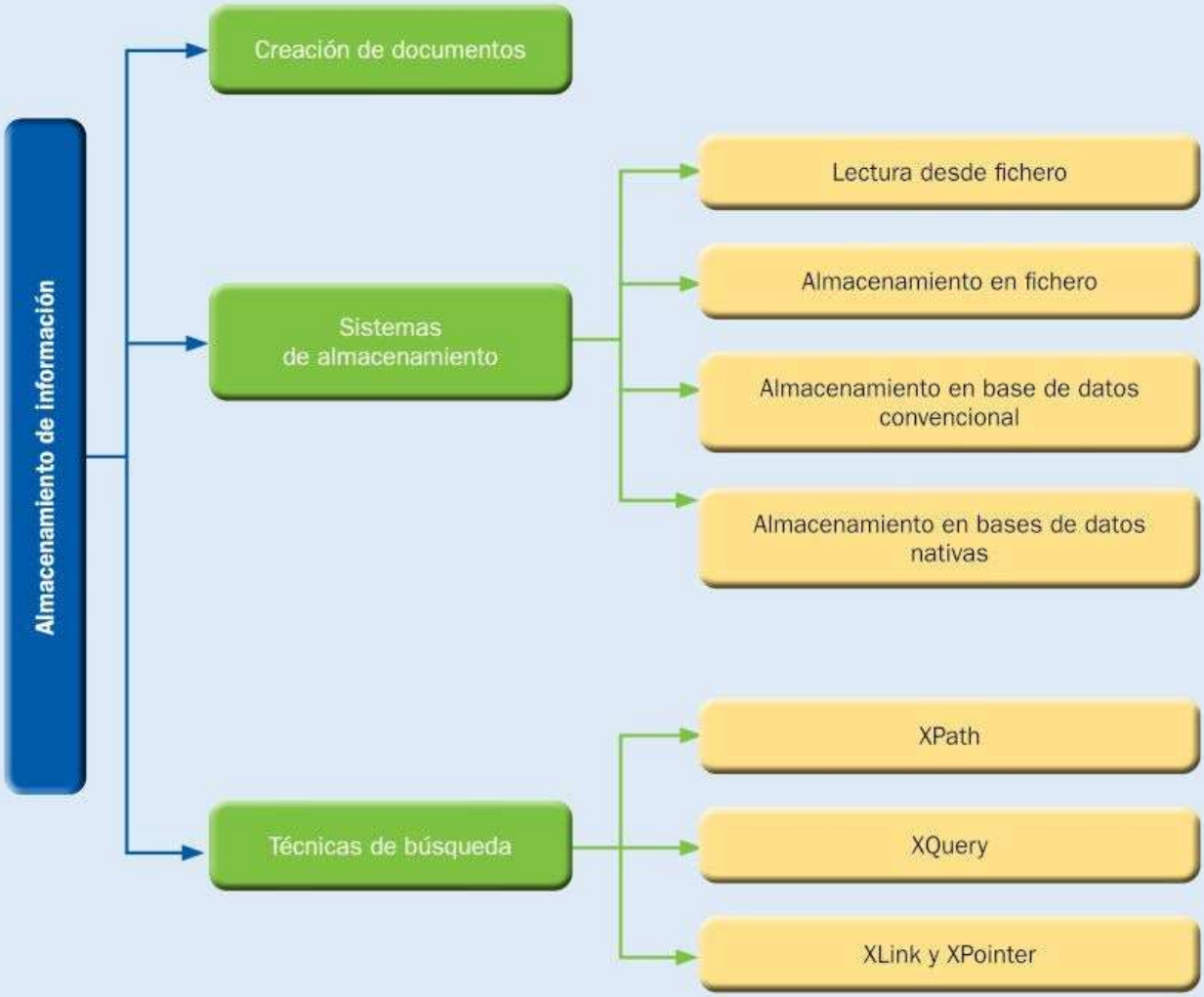
En el ejemplo siguiente se puede observar la sintaxis de una expresión XLink utilizando XPointer para precisar el destino del enlace: el atributo **href** marca dicho destino; el símbolo # indica, utilizando XPointer, que el enlace se debe realizar al elemento cuyo atributo **id** tenga como valor **Estepona.**

```
<destinos xmlns:xlink="http://www.w3.org/1999/xlink">
<direccion xlink:type="simple" xlink:href="http://localhost/destinos.
xml#Estepona">Playa de Estepona</direccion>
```

XLink es el lenguaje de enlace XML, mientras que XPointer es el lenguaje de direccionamiento.

© Ediciones Paraninfo

MAPA CONCEPTUAL



© Ediciones Paraninfo

Actividades de comprobación

5.1. ¿Qué significan las siglas JAXP?

a) *Java API for eXternal Processing.*
b) *Java Advanced XML Processor.*
c) *Java API for XML Processing.*
d) *Java Application for XML Processor.*

5.2. ¿Qué significan las siglas SAX?

a) *Simple API for XML.*
b) *EaSy API for XML.*
c) *Super API for XML.*
d) *Slow API for XML.*

5.3. ¿Qué significan las siglas DOM?

a) *Document Oriented Model.*
b) *Domain Object Model.*
c) *Document Object Model.*
d) *Domain Oriented Model.*

5.4. ¿Cuál de las siguientes es una característica de SAX?

a) Ocupa más en memoria.
b) Ocupa menos en memoria.
c) Es más lento.
d) Es más versátil.

5.5. ¿Cuál de las siguientes es una característica de DOM?

a) Ocupa menos en memoria.
b) El documento se procesa según se lee.
c) Es más versátil.
d) Solo permite realizar lecturas en una dirección.

5.6. ¿Cuál es la secuencia de pasos habitual en el procesado de un documento XML?

a) Almacenamiento-creación-recuperación-procesado.
b) Recuperación-creación-almacenamiento-procesado.
c) Creación-almacenamiento-recuperación-procesado.
d) Procesado-almacenamiento-recuperación-creación.

5.7. ¿Qué solución tecnológica para el almacenamiento de XML es más independiente?

a) Bases de datos relacionales.
b) Bases de datos XML.
c) Ficheros.
d) Todas las soluciones tienen el mismo nivel de dependencia.

5.8. ¿Cómo se llama la clase que almacena en memoria un documento XML en Java?

a) Element.
b) Node.
c) Document.
d) Tree.

© Ediciones Paraninfo

5.9. ¿Cómo se puede almacenar un documento XML en una base de datos relacional?

a) Como un campo numérico.
b) Como un campo de texto.
c) Como un campo lógico.
d) Todas las respuestas son correctas.

5.10. ¿Cuál de las siguientes características no es propia de BaseX?

a) Es un gestor de base de datos XML.
b) Admite expresiones XPath.
c) Es un lenguaje de programación basado en XML.
d) Tiene diferentes vistas para mostrar los documentos XML.

5.11. ¿Qué proporciona la expresión XPath //personaje?

a) Todos los elementos <personaje> que son hijos del elemento raíz.
b) Todos los elementos <personaje> independientemente de su ubicación.
c) El primer elemento <personaje> del documento.
d) Un valor booleano indicando si existe un elemento <personaje>.

5.12. ¿Qué proporciona la expresión XPath //personaje/nombre?

a) El primer elemento <nombre> que es hijo de un elemento <personaje>.
b) El elemento <personaje> que tiene como hijo un elemento <nombre>.
c) Todos los elementos <nombre> que son hijos de un elemento <personaje>.
d) Todos los elementos <personaje> y <nombre>.

5.13. ¿Qué significa en XPath la expresión //@*?

a) Todos los nodos.
b) Todos los atributos de todos los nodos.
c) Todos los comentarios de todos los nodos.
d) Todos los textos contenidos en todos los nodos.

5.14. Indica cuál de las siguientes opciones no se corresponde con una tecnología XML:

a) XLink.
b) XPointer.
c) XReference.
d) XQuery.

Actividades de aplicación

5.15. Explica cuándo y por qué es mejor utilizar el analizador SAX. Pon un ejemplo.

5.16. ¿Cuándo y por qué es mejor utilizar el analizador DOM? Pon un ejemplo.

5.17. Si tienes que procesar un documento XML que contiene una lista con información sobre canales de YouTube y quieres enviar un correo electrónico por cada canal, ¿utilizarías SAX o DOM? Justifica la respuesta.

© Ediciones Paraninfo

5.18. Si tienes que procesar un documento XML que contiene una lista con información sobre canales de YouTube y quieres enviar un correo electrónico para el canal elegido por el usuario, ¿utilizarías SAX o DOM? Justifica la respuesta.

5.19. Si tienes un documento XML con una enorme cantidad de información, ¿qué analizador sería tu primer candidato? Justifica la respuesta.

5.20. Utilizando el lenguaje de programación que prefieras, crea un documento XML y guárdalo en un fichero.

5.21. Realiza un documento XML que contenga información sobre recetas de cocina, incluyendo un dato referente al nivel de dificultad (fácil, medio, avanzado). Crea una expresión XPath que muestre las recetas de nivel fácil.

5.22. Crea un documento XML que contenga información sobre grupos musicales, incluyendo un dato referente al género (rock, punk, pop, heavy…). Crea una expresión XPath que muestre los nombres de los grupos del género rock.

5.23. Crea un documento XML que contenga información sobre países. Incluye el número de habitantes en un atributo. Crea una expresión XPath que muestre todos los datos de los países con más de diez millones de habitantes.

5.24. Realiza un documento XML que contenga información sobre países. Incluye el número de habitantes en un atributo. Crea una expresión XPath que muestre solo los nombres de los países con menos de diez millones de habitantes.

5.25. Utilizando XQuery escribe una consulta que devuelva todos los elementos <modelo> que haya en un documento almacenado en el fichero «datos.xml».

5.26. Haciendo uso de XQuery escribe una consulta que devuelva todos los elementos <modelo> cuyo atributo «velocidad» tenga un valor mayor que 1000 y que se encuentren en un fichero con el nombre «datos.xml».

Actividades de ampliación

5.27. Amplía tus conocimientos sobre el analizador SAX.

5.28. Obtén más información sobre el analizador DOM.

5.29. Busca información referente a StAX. Averigua en qué se diferencia de SAX y de DOM.

5.30. Halla información referente a cómo MySQL soporta documentos XML y XPath.

5.31. Busca información relacionada con cómo Oracle Database soporta documentos XML y XPath.

5.32. Investiga sobre las distintas vistas de los documentos XML que proporciona BaseX.

© Ediciones Paraninfo

Enlaces web de interés

- **JAVA SAX** - https://docs.oracle.com/javase/tutorial/jaxp/sax/index.html
  (Información sobre el analizador SAX para Java)

- **JAVA DOM** - https://docs.oracle.com/javase/tutorial/jaxp/dom/index.html
  (Información sobre el analizador DOM para Java)

- **BASE X** - https://basex.org/
  (Portal web sobre la base de datos BaseX)

- **XPath en MDN** - https://developer.mozilla.org/es/docs/Web/XPath
  (Información sobre XPath en MDN)

- **FLWOR** - https://docs.microsoft.com/es-es/sql/xquery/flwor-statement-and-iteration-xquery?view=sql-server-ver15
  (Información sobre XQuery y FLWOR en la web de Microsoft en el contexto de SQLServer)

- **XLink y XPointer** - https://www.w3schools.com/xml/xml_xlink.asp
  (Información sobre XLink y XPointer en la web de w3schools)

© Ediciones Paraninfo

UNIDAD 6

Conversión y adaptación de documentos XML

Objetivos

- Explicar el significado de la familia de lenguajes XSL.
- Conocer el funcionamiento de las transformaciones XSLT.
- Detallar el proceso de adaptación de los datos a los distintos formatos de salida.
- Construir documentos físicos mediante el uso de procesadores.
- Presentar varias alternativas de software para realizar las transformaciones.

Contenidos

- 6.1. La familia de lenguajes de hojas de estilo extensibles: XSL
- 6.2. XSLT
- 6.3. XSL-FO
- 6.4. Herramientas de transformación XML

Introducción

En el ecosistema del metalenguaje de marcas XML existen tecnologías que permiten transformar los datos contenidos en los documentos escritos con él.

Esta transformación tiene dos vías principales: la transformación para el cambio de lenguaje de representación (por ejemplo, de XML a HTML) y la adaptación para su presentación en diversos dispositivos o formatos físicos de salida.

XSL es el paraguas que cobija a estas tecnologías: XSLT, XSL-FO y XPath. Estas tecnologías se tratarán en esta unidad, a excepción de XPath que se presentó en detalle en la Unidad 5.

6.1. La familia de lenguajes de hojas de estilo extensibles: XSL

XSL (*eXtensible Stylesheet Language,* «lenguajes de hojas de estilo extensible», en castellano) es una familia de tecnologías desarrolladas por el W3C que tienen como objetivo proporcionar herramientas para transformar los documentos XML.

Una de las preguntas que surge de forma inmediata es en qué se puede transformar un documento XML. La respuesta es simple y a la vez sorprendente: en cualquier cosa que se pueda representar mediante cadenas de caracteres. Es decir, se trata de una importante lista de alternativas, ya que son muchas las tecnologías que expresan sus datos mediante cadenas de caracteres: el lenguaje HTML, el formato CSV, las sentencias SQL, los programas informáticos, las imágenes vectoriales y cualquier dialecto de XML, por citar algunos ejemplos.

Nota técnica

El formato CSV representa los datos en texto plano separado por comas. Es un formato muy utilizado por su sencillez.

La familia XSL está formada por tres elementos: XSLT *(XSL Transformation),* XSL-FO *(XSL Formatting Objects)* y XPath *(XML Path Language).*

XSLT es el lenguaje de transformación propiamente dicho. En muchas ocasiones XSL y XSLT hacen referencia al mismo concepto: la transformación de documentos XML.

XSL-FO es el lenguaje en el que se indica el formato que debe tener un documento XML para representarse en diferentes dispositivos o medios, como pantallas o papel.

XPath es un lenguaje que permite recorrer un documento XML utilizando expresiones. Se utiliza en XSLT para seleccionar las diferentes partes de los documentos.

En esta unidad se abordan en profundidad los lenguajes XSLT y XSL-FO.

© Ediciones Paraninfo

Recuerda

El lenguaje XPath se trata en este mismo libro en la Unidad 5, correspondiente al almacenamiento de información en los lenguajes de marcas.

6.2. XSLT

Un documento XML es un contenedor de datos estructurados, formado por elementos, atributos y dependencias jerárquicas. Estos datos están delimitados por unas etiquetas o marcas que no tienen más significado que el que les quiera asignar el creador del documento. En HTML las etiquetas están predefinidas en la especificación del lenguaje y tienen un significado asociado con un estilo de presentación (propio o asignado mediante CSS).

Con XSLT se puede transformar un documento expresado en el formato XML en un documento equivalente expresado en otro formato, normalmente HTML.

6.2.1. Estructura

El proceso de transformación consiste en seleccionar diferentes partes del documento XML de entrada e indicar en qué se quieren transformar. La selección se realiza mediante XPath y la transformación mediante XSLT. Las transformaciones XSLT se almacenan en **ficheros XML con extensión .xsl.**

Los componentes que forman parte de una transformación son los siguientes:

- Un documento XML de entrada.
- Un fichero de transformación XSLT.
- Un procesador con capacidad para aplicar las transformaciones (un editor o un navegador web).

Nota técnica

Las políticas de seguridad de los navegadores web pueden bloquear el funcionamiento de las transformaciones XSLT al abrir el documento XML desde el sistema de archivos. Para evitar este problema existen dos alternativas:

- Alojar el documento XML y el fichero .xsl en un servidor web (local o remoto) y acceder al documento XML mediante el protocolo HTTP.
- Modificar la configuración de seguridad del navegador para evitar el bloqueo.

En el navegador Firefox la modificación de la configuración de seguridad se realiza siguiendo los siguientes pasos:

- Abrir el navegador Firefox.
- Escribir en la barra de direcciones **about:config.**
- Pulsar sobre «Aceptar el riesgo y continuar».
- Buscar el parámetro **privacy.file_unique_origin** y cambiar el valor a **false.**

© Ediciones Paraninfo

El documento XML debe incluir una referencia al documento XSL:

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="ejemplo1.xsl"?>
…
```

La estructura del documento XSL se muestra a continuación. Debe incluir obligatoriamente la declaración del espacio de nombres de XSL y, habitualmente, el elemento **<template match="/">**, que indica que se debe aplicar la transformación a todo el documento XML.

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/
Transform">
<xsl:template match="/">
…
</xsl:template>
</xsl:stylesheet>
```

6.2.2. Transformación

Una vez presentada la estructura básica de los elementos que participan en el proceso de transformación, hay que definir dicho proceso.

En el documento XSL se alternan **texto libre** (normalmente HTML) con **elementos XSL.**

Los textos libres se volcarán en la salida sin sufrir ninguna modificación.

El siguiente documento XSL aplicado a cualquier documento XML generará una salida consistente en el código HTML incluido dentro de la etiqueta <template>.

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/
Transform">
<xsl:template match="/">
  <html>
  <body>
    <p>Un texto marcado como párrafo</p>
  </body>
  </html>
</xsl:template>
</xsl:stylesheet>
```

Los elementos XSL son los que aportan las reglas de transformación que modifican los datos contenidos en el documento XML. En el siguiente ejemplo, incrustado en el código HTML se encuentra un elemento XSL cuyo significado es «muestra el valor del elemento <titulo> que se encuentra dentro del elemento <pelicula> del documento XML de entrada».

© Ediciones Paraninfo

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/
Transform">
<xsl:template match="/">
  <html>
  <body>
    <p><xsl:value-of select="pelicula/titulo"/></p>
  </body>
  </html>
</xsl:template>
</xsl:stylesheet>
```

En el siguiente ejemplo, más completo, se muestran el fichero XML de entrada, el fichero XSL de transformación y el resultado proporcionado por el navegador.

Fichero XML de entrada: ejemplo1.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="ejemplo1.xsl"?>
<pelicula>
  <titulo>Superman: la película</titulo>
  <argumento>Un extraterrestre con increíbles poderes se dedica a salvar a
los habitantes del planeta Tierra</argumento>
</pelicula>
```

Fichero XSL con las reglas de transformación: ejemplo1.xsl

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:template match="/">
    <html lang="en">
      <head>
        <meta charset="UTF-8"/>
      </head>
      <body>
        <h2>
          <xsl:value-of select="pelicula/titulo"/>
        </h2>
        <p>
          <xsl:value-of select="pelicula/argumento"/>
        </p>
      </body>
    </html>
  </xsl:template>
</xsl:stylesheet>
```

© Ediciones Paraninfo

Resultado proporcionado por el navegador:

Figura 6.1. *El navegador muestra el resultado de la transformación.*

Como se puede observar, al abrir desde el navegador web (Firefox en este ejemplo), en el documento XML se realizan automáticamente las transformaciones declaradas en el fichero de transformación. Este contiene, en este caso, código HTML convencional junto con instrucciones XSLT, generando una vista equivalente a una página web convencional.

Los elementos específicos de XSL utilizados en este ejemplo se explican a continuación.

El elemento siguiente indica que nos encontramos ante una transformación y establece **xsl** como prefijo para las etiquetas del espacio de nombres de XSL:

```
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
```

El elemento <xsl:template> se utiliza para crear una plantilla relacionada con un atributo de coincidencia. En este caso, al utilizar el valor "/" se está indicando que la plantilla afecta a todo el documento:

```
<xsl:template match="/">
```

Los elementos <xsl:value-of> indican que deben ser sustituidos por el valor del elemento obtenido como consecuencia de la consulta realizada con XPath y establecida en el atributo **select.**

```
<xsl:value-of select="pelicula/titulo"/>
```

```
<xsl:value-of select="pelicula/argumento"/>
```

El resto es código HTML convencional dentro del cual se incrustan los valores proporcionados por los elementos <xsl:value-of>, dando lugar a un documento HTML.

© Ediciones Paraninfo

Hay que considerar que la transformación no tiene por qué ser necesariamente a código HTML, pese a que esta es la operación habitual. Por ejemplo, se puede realizar una transformación que genere un documento JSON:

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/
Transform">
<xsl:output method="text"/>
<xsl:template match="/">
{
    "titulo":"<xsl:value-of select="pelicula/titulo"/>",
    "argumento":"<xsl:value-of select="pelicula/argumento"/>"
}
</xsl:template>
</xsl:stylesheet>
```

El elemento **<xsl:output>** determina el formato del documento que se va a generar. Al indicar que es tipo "**text**" se respetan los saltos de línea y las tabulaciones.

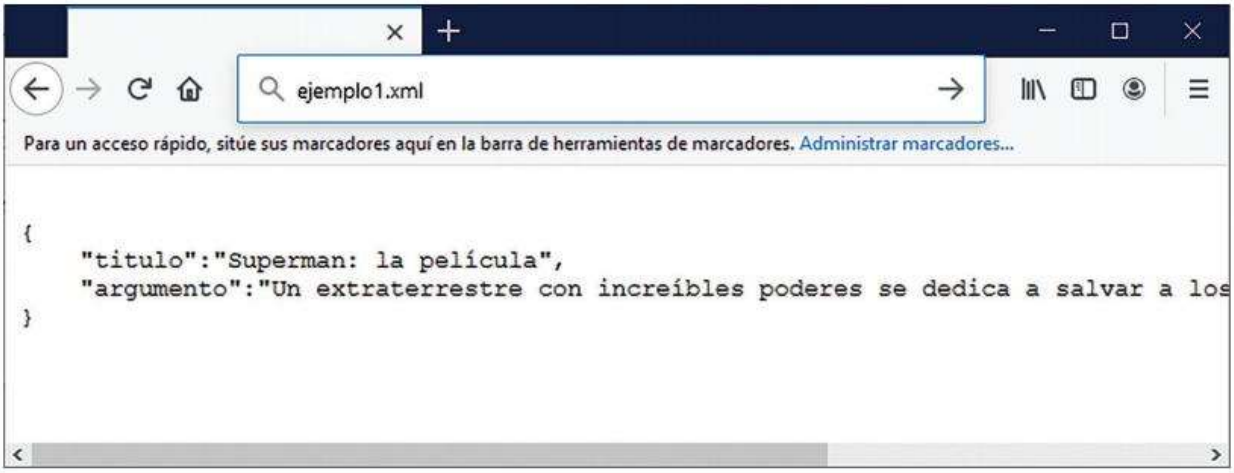


Figura 6.2. *La salida de la transformación es una cadena de caracteres.*

También se podría crear un generador de código Java con XSLT. En la siguiente transformación se genera la declaración y la instanciación de un objeto:

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/
Transform">
<xsl:output method="text"/>
<xsl:template match="/">
Pelicula p = new Pelicula("<xsl:value-of select="pelicula/titulo"/>",
"<xsl:value-of select="pelicula/argumento"/>");
</xsl:template>
</xsl:stylesheet>
```

© Ediciones Paraninfo

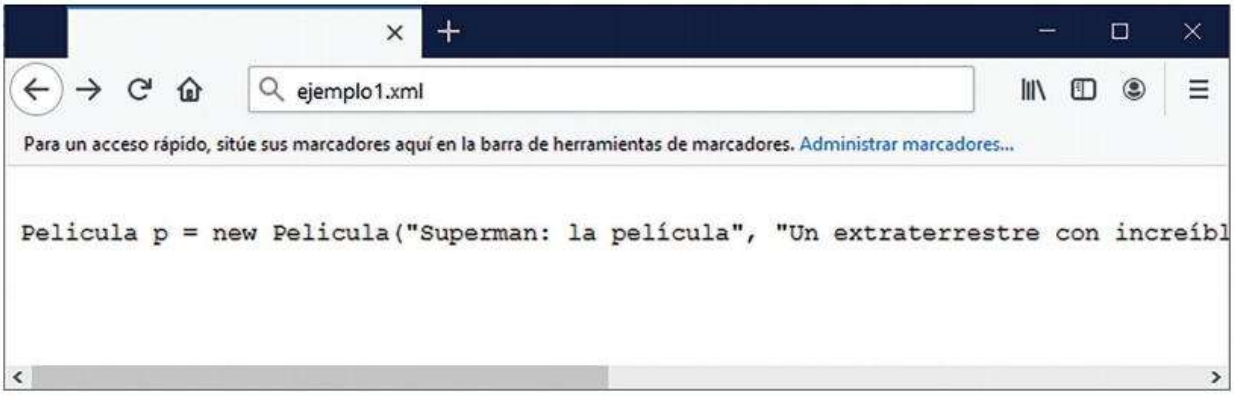


Figura 6.3. *La transformación puede generar código fuente a partir de los datos XML.*

En resumen, con XSLT se puede generar cualquier salida en formato texto a partir de un documento XML.

6.2.3. Elementos de XSLT

XSLT dispone de un extenso número de elementos para crear las transformaciones. En la Tabla 6.1 se describen los más importantes.

Tabla 6.1. Elementos de XSLT

Elemento	Sintaxis	Descripción
<xsl:value-of>	<xsl:value-of select="*expresión XPath*"/>	Obtiene el valor de un elemento o de un atributo del documento XML.
<xsl:for-each>	<xsl:for-each select="*expresión XPath*">	Recorre un conjunto de elementos.
<xsl:sort>	<xsl:sort select="*nombre del elemento por el que se quiere ordenar*"/>	Ordena un conjunto de elementos.
<xsl:if>	<xsl:if test="*expresión*">	Permite condicionar la transformación en función de una condición.
<xsl:choose>	<xsl:choose> <xsl:when test="*expresión*"> … </xsl:when> <xsl:otherwise> … </xsl:otherwise> </xsl:choose>	Utilizado conjuntamente con **<xsl:when>** y **<xsl:otherwise>,** permite condicionar la transformación en función de múltiples condiciones.
<xsl:template>	<xsl:template match="*expresion XPath*">	Permite establecer una plantilla y determinar sobre qué conjunto de elementos se va a realizar la transformación.
<xsl:output>	<xsl:ouput *parametros*>	Define el formato de salida del documento.
<xsl:comment>	<xsl:comment> Comentario </xsl:comment>	Delimita un comentario. El contenido no será procesado.

© Ediciones Paraninfo

A continuación, se muestran ejemplos de aplicación de algunos de estos elementos sobre el siguiente documento XML de entrada:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<?xml-stylesheet type="text/xsl" href="ejemplo_elementos.xsl"?>
<gastos>
    <mes>Septiembre</mes>
    <gasto desgravable="si">
        <concepto>Materail fungible</concepto>
        <importe>450</importe>
    </gasto>
    <gasto desgravable="no">
        <concepto>Gastos sin justificar</concepto>
        <importe>250</importe>
    </gasto>
    <gasto desgravable="si">
        <concepto>Transporte</concepto>
        <importe>300</importe>
    </gasto>
</gastos>
```

Elemento <value-of>: muestra el contenido del elemento <mes>.

```
<xsl:value-of select="//mes"/>
```

Elemento <for-each>: muestra todos los elementos <concepto> que son hijos de <gasto>.

```
<xsl:for-each select="//gasto/concepto">
   <xsl:value-of select="."/>
</xsl:for-each>
```

Elemento <sort>: ordena los elementos <gasto> del documento XML por el elemento <importe> contenido en dichos elementos.

```
<xsl:for-each select="//gasto">
   <xsl:sort select="importe" data-type="number"/>
      <xsl:value-of select="concepto"/>
</xsl:for-each>
```

Elemento <if>: muestra solo los elementos <concepto> que son descendientes de los elementos <gasto> del XML si el atributo «desgravable» de estos últimos tienen el valor «si».

```
<xsl:for-each select="//gasto">
   <xsl:if test="./@desgravable='si'">
      <xsl:value-of select="concepto"/>
   </xsl:if>
</xsl:for-each>
```

© Ediciones Paraninfo

Elementos <choose>, <when> y <otherwise>: muestran un contenido u otro en función del valor del atributo «desgravable» de los elementos <gasto> del documento XML.

```
<xsl:for-each select="//gasto">
      <xsl:choose>
         <xsl:when test="./@desgravable='si'">
            <xsl:value-of select="concepto"/> es desgravable<br/>
         </xsl:when>
      <xsl:otherwise>
         <xsl:value-of select="concepto"/> no es desgravable<br/>
      </xsl:otherwise>
      </xsl:choose>
</xsl:for-each>
```

Nota técnica

La lista completa de elementos se puede consultar en el siguiente enlace:

https://developer.mozilla.org/es/docs/Web/XSLT/Element

Actividad resuelta 6.1

Generación de una tabla con XSL

En esta actividad, dado un documento XML de entrada, se genera mediante una transformación XSLT una tabla a la que se le aplica un estilo CSS.

En la tabla se muestran, para todas las obras creadas a partir de 1850, el país, el año de creación, el título, el autor y el museo en el que están expuestas.

El documento XML de entrada es el siguiente:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<?xml-stylesheet type="text/xsl" href="transformacion.xsl"?>
<obras>
    <obra pais="Francia" anyo="1876">
        <titulo>Bal au moulin de la Galette</titulo>
        <autor>Pierre-Auguste Renoir</autor>
        <museo visible="true">Museo de Orsay</museo>
    </obra>
    <obra pais="Francia" anyo="1879">
        <titulo>Pont de Maincy</titulo>
        <autor>Paul Cézanne</autor>
        <museo visible="false">Museo de Orsay</museo>
    </obra>
    <obra pais="España" anyo="1822">
        <titulo>Amalia de Llano y Dotres, condesa de Vilches</titulo>
        <autor>Federico de Madrazo y Kuntz</autor>
        <museo visible="true">Museo del Prado</museo>
   </obra>
    <obra pais="España" anyo="1823">
       <titulo>Saturno devorando a su hijo</titulo>
```


© Ediciones Paraninfo

```
        <autor>Francisco José de Goya y Lucientes</autor>
        <museo visible="true">Museo del Prado</museo>
    </obra>
    <obra pais="España" anyo="1635">
        <titulo>El príncipe Baltasar Carlos a caballo</titulo>
        <autor>Diego Rodríguez de Silva y Velázquez</autor>
        <museo visible="false">Museo del Prado</museo>
    </obra>
    <obra pais="Países Bajos" anyo="1890">
        <titulo>Trigal con cuervos</titulo>
        <autor>Vincent Willem van Gogh</autor>
        <museo visible="true">Museo Van Gogh</museo>
    </obra>
</obras>
```

Recuerda

Hay que incluir en el documento XML la referencia al fichero XSL que contiene la información sobre la transformación que se va a realizar.

Solución

El documento de transformación, contenido en el fichero **transformacion.xsl**, es el siguiente:

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
  <html>
  <style>
    th {
      background-color:#1E7B05;
      color:white;
      padding:0.5em;
    }
    td {
      padding:0.5em;
    }
    tr:nth-child(even) {
      background: #CCC
    }
    tr:nth-child(odd) {
      background: #FFF
    }
    .blueColor {
      color:blue;
    }
  </style>
  <body>
    <table border="1">
      <tr>
        <th>País</th>
        <th>Año</th>
        <th>Título</th>
       <th>Autor</th>
       <th>Museo</th>
```

© Ediciones Paraninfo

```
      </tr>
      <xsl:for-each select="obras/obra">
        <xsl:sort select="titulo"/>
          <xsl:if test="@anyo &gt; 1850">
            <tr>
              <td><xsl:value-of select="@pais"/></td>
              <td><xsl:value-of select="@anyo"/></td>
              <td><xsl:value-of select="titulo"/></td>
              <td><xsl:value-of select="autor"/></td>
              <td><xsl:value-of select="museo"/></td>
            </tr>
          </xsl:if>
      </xsl:for-each>
    </table>
  </body>
  </html>
</xsl:template>
</xsl:stylesheet>
```

En negrita se resaltan las instrucciones XSLT, que en conjunción con las expresiones XPath utilizadas como selector permiten seleccionar las distintas partes del documento de entrada y determinar qué valores deben generarse como salida. El resto está compuesto por texto HTML corriente que se utilizará en la generación de la salida sin sufrir modificaciones.

Las sentencias relevantes de dicho fichero XSL son las siguientes:

Declaración del espacio de nombres:

```
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
```

Establecimiento de la plantilla de transformación. En este caso, se partirá del elemento raíz del documento XML de entrada.

```
<xsl:template match="/">
```

Elemento for-each que recorrerá todos los elementos **obra** contenidos en el elemento **obra.**

```
<xsl:for-each select="obras/obra">
```

Sobre la lista de elementos **obra** que se están recorriendo, se realiza una ordenación por el elemento **titulo.**

```
<xsl:sort select="titulo"/>
```

Se filtran los elementos **obra** haciendo que se genere una fila HTML solo para aquellos elementos cuyo atributo **@anyo** sea mayor que 1850.

```
<xsl:if test="@anyo &gt; 1850">
```

Se muestran los valores (atributos precedidos del símbolo @ y elementos indicando su nombre) de cada elemento seleccionado. Estos elementos están rodeados por las etiquetas <td> y </td> de HTML para formar las distintas celdas de la tabla.

```
<td><xsl:value-of select="@anyo"/></td>
<td><xsl:value-of select="titulo"/></td>
```

Como se puede observar en la Figura 6.4, los datos de entrada (fichero XML) han sido transformados en una tabla HTML a la que se le ha aplicado un estilo CSS incluido en el fichero de transformación XSL, tal y como se solicita en el enunciado de esta actividad.

El resultado generado por el navegador web al acceder al documento XML es el siguiente:

© Ediciones Paraninfo

País	Año	Título	Autor	Museo
Francia	1876	Bal au moulin de la Galette	Pierre-Auguste Renoir	Museo de Orsay
Francia	1879	Pont de Maincy	Paul Cézanne	Museo de Orsay
Países Bajos	1890	Trigal con cuervos	Vincent Willem van Gogh	Museo Van Gogh

Figura 6.4. *XSL permite transformar un documento XML en código HTML.*

Actividades propuestas

6.1. Generación de la ficha de un vehículo

Utilizando XSLT, crea un documento HTML en el que se muestre una ficha con los datos de un vehículo a partir del documento XML siguiente:

```
<?xml version="1.0" encoding="UTF-8"?>
<vehiculo>
  <marca>Ferrari</marca>
  <web>https://www.ferrari.com/</web>
  <modelo>F8 Spider</modelo>
  <precio>262000</precio>
</pelicula>
```

6.2. Creación de una lista de ingredientes

Utilizando XSLT, crea un documento HTML en el que se muestre una lista con los ingredientes de una receta a partir del documento XML siguiente:

```
<?xml version="1.0" encoding="UTF-8"?>
<receta nombre="Fabada Asturiana">
  <ingrediente nombre="Fabes"/>
  <ingrediente nombre="Panceta curada asturiana"/>
  <ingrediente nombre="Chorizo asturiano"/>
  <ingrediente nombre="Morcilla asturiana"/>
  <ingrediente nombre="Hueso de jamón"/>
  <ingrediente nombre="Azafrán"/>
</receta>
```

6.3. Creación de un *curriculum vitae*

Utilizando XSLT, crea un documento HTML en el que aparezcan los datos de tu currículum. Diseña el documento XML de entrada teniendo en cuenta que debe incluir los siguientes datos:

- Nombre completo.
- Número de teléfono y dirección de correo electrónico.
- Formación académica, incluyendo nombre del título o curso, centro donde se obtuvo y año de finalización.
- Experiencia profesional, incluyendo nombre del puesto, nombre de la empresa, año de comienzo y año de finalización.

Utiliza estilos CSS para conseguir que el currículum sea atractivo visualmente.

© Ediciones Paraninfo

6.3. XSL-FO

XSL-FO *(Extensible Stylesheet Language Formatting Objects)* es un lenguaje de marcas basado en XML que forma parte de la familia de lenguajes de XSL. Permite describir el modo de mostrar un conjunto de datos en un soporte determinado, ya sea este papel, la pantalla de un ordenador u otros medios alternativos. En los objetos XSL-FO se definen aspectos como las características de la página (dimensiones, márgenes...), párrafos, listas, tablas, etcétera.

Para su uso es necesario disponer de un programa que sea capaz de procesar XSL-FO.

Con XSL-FO se pueden crear documentos en diversos formatos, como PS *(PostScript),* RTF *(Rich Text format),* PNG *(Portable Network Graphics),* PCL *(Printer Command Language)* o PDF *(Portable Document Format),* por citar algunos ejemplos.

Los ficheros XLS-FO tienen la extensión .fo y .xml.

6.3.1. Fundamentos

Existen dos alternativas a la hora de utilizar XSL-FO: con los datos embebidos en el propio documento XSL-FO u obteniéndolos de un documento XML.

En ambos casos es necesario disponer de una aplicación que sea capaz de procesar la información contenida en un documento XSL-FO. La más utilizada es Apache FOP *(Formatting Objects Processor),* con licencia de software libre y desarrollada en Java.

Nota técnica

Puedes descargar e instalar Apache FOP desde el siguiente enlace:

https://xmlgraphics.apache.org/fop/

Datos embebidos en el documento

A continuación, se muestra un ejemplo de XSL-FO con los datos incrustados en el documento, que se almacena con el nombre **ejemplo1.fo:**

```
<?xml version="1.0" encoding="utf-8"?>
<fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">
  <fo:layout-master-set>
    <fo:simple-page-master master-name="A4" page-height="29.7cm" page-
width="21cm" margin-top="1cm" margin-bottom="1cm" margin-left="2cm" margin-
right="2cm">
      <fo:region-body margin-top="1cm"/>
      <fo:region-before extent="3cm"/>
      <fo:region-after extent="1.5cm"/>
```


© Ediciones Paraninfo

```
    </fo:simple-page-master>
  </fo:layout-master-set>
  <fo:page-sequence master-reference="A4">
    <fo:flow flow-name="xsl-region-body">
      <fo:block font-size="16pt" font-family="sans-serif" line-height="30pt"
space-after.optimum="15pt" background-color="#444444" color="white" text-
align="center" padding-top="2pt">
        La vida del Buscón
      </fo:block>
      <fo:block font-size="10pt" font-family="sans-serif" line-height="15pt"
space-after.optimum="3pt" text-align="justify">
Yo, señora, soy de Segovia. Mi padre se llamó Clemente Pablo, natural del
mismo pueblo; Dios le tenga en el cielo. Fue, tal como todos dicen, de oficio
barbero, aunque eran tan altos sus pensamientos que se corría de que le
llamasen así, diciendo que él era tundidor de mejillas y sastre de barbas.
      </fo:block>
    </fo:flow>
  </fo:page-sequence>
</fo:root>
```

El proceso de conversión en un documento es muy sencillo, una vez se tiene instalado el programa adecuado. Con Apache FOP, basta con ejecutar el programa **fop** desde un terminal del sistema. La llamada tiene esta sintaxis:

```
fop fichero_fo -tipo_salida fichero_salida
```

La siguiente ejecución de **fop** genera un documento PDF que toma como origen el fichero **ejemplo1.fo** y nombra la salida como **ejemplo1.pdf:**

```
fop ejemplo1.fo -pdf ejemplo1.pdf
```

El resultado es el que recoge la Figura 6.5.

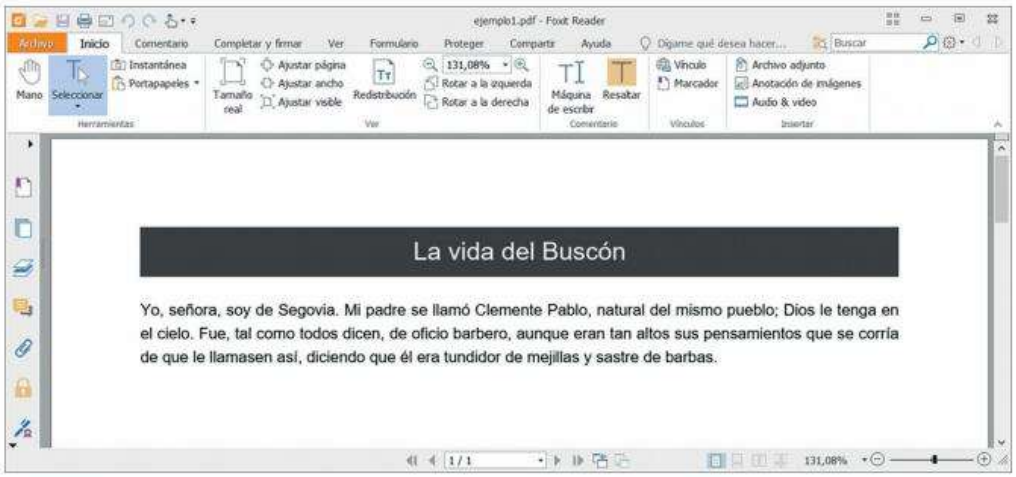


Figura 6.5. *XSL-FO se utiliza a menudo para generar documentos en formato PDF.*

© Ediciones Paraninfo

Como ya se ha indicado, existen más alternativas de formato de salida, como PostScript (PS):

```
fop ejemplo1.fo -ps ejemplo1.ps
```

Rich Text Format (RTF):

```
fop ejemplo1.fo -rtf ejemplo1.rtf
```

o Portable Network Graphics (PNG):

```
fop ejemplo1.fo -png ejemplo1.png
```

Datos obtenidos de un documento XML

Cuando la fuente de los datos es un documento XML externo, XSL-FO se debe incluir dentro de una hoja de transformación XSLT. La sintaxis es la misma, pero, en lugar del texto incluido en la solución de los datos embebidos, se utilizan expresiones XSLT para seleccionar las partes del documento XML que se quieren mostrar.

El ejemplo equivalente a la solución embebida está formado por un documento XML de entrada (**ejemplo2.xml**) y un documento XLST (**ejemplo2.xsl**) que contiene los elementos XLS-FO.

El código del documento XML es el que se muestra a continuación. Como se puede observar, la estructura es la habitual y no contiene ninguna referencia a XSL-FO ni a XSLT.

```
<?xml version="1.0" encoding="utf-8"?>
<obra>
    <titulo>El Monte de las Ánimas</titulo>
    <texto>La noche de difuntos me despertó, a no sé qué hora, el doble
de las campanas; su tañido monótono y eterno me trajo a las mientes esta
tradición que oí hace poco en Soria.</texto>
</obra>
```

El contenido del documento XSLT es el siguiente:

```
<?xml version="1.0" encoding="utf-8"?>
<xsl:stylesheet version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  xmlns:fo="http://www.w3.org/1999/XSL/Format">
  <xsl:output method="xml" indent="yes"/>
  <xsl:template match="/">
    <fo:root>
      <fo:layout-master-set>
```


© Ediciones Paraninfo

```
        <fo:simple-page-master master-name="A4" page-height="29.7cm" page-
width="21cm" margin-top="1cm" margin-bottom="1cm" margin-left="2cm" margin-
right="2cm">
          <fo:region-body margin-top="1cm"/>
          <fo:region-before extent="3cm"/>
          <fo:region-after extent="1.5cm"/>
        </fo:simple-page-master>
      </fo:layout-master-set>
      <fo:page-sequence master-reference="A4">
        <fo:flow flow-name="xsl-region-body">
          <fo:block font-size="16pt" font-family="sans-serif" line-
height="30pt" space-after.optimum="15pt" background-color="#444444"
color="white" text-align="center" padding-top="2pt">
            <xsl:value-of select="/obra/titulo"/>
          </fo:block>
          <fo:block font-size="10pt" font-family="sans-serif" line-
height="15pt" space-after.optimum="3pt" text-align="justify">
            <xsl:value-of select="/obra/texto"/>
          </fo:block>
        </fo:flow>
      </fo:page-sequence>
    </fo:root>
  </xsl:template>
</xsl:stylesheet>
```

Como se puede observar, el contenido es prácticamente idéntico al de la solución embebida, con la salvedad de que se trata de una hoja de transformación (de ahí que se utilicen elementos del espacio de nombres con el prefijo **xsl** como contenedores principales) y que donde antes se incluía el texto que mostrar ahora se incluyen expresiones XSLT.

Recuerda

XSLT y XSL-FO forman parte de la misma familia de lenguajes y se complementan, pero no son lo mismo. XSLT es un lenguaje relativo a las transformaciones y XSL-FO es un lenguaje relativo al formato.

La generación del documento también se realiza de forma diferente, ya que ahora participan más ficheros. La sintaxis es la siguiente:

```
fop -xml fichero_xml -xsl fichero_xsl tipo_salida fichero_salida
```

La generación, en este ejemplo, se realiza ejecutando **fop** con la siguiente invocación desde un terminal:

```
fop -xml ejemplo2.xml -xsl ejemplo2.xsl -pdf ejemplo2.pdf
```

© Ediciones Paraninfo

El resultado obtenido se puede observar en la captura de pantalla de la Figura 6.6.

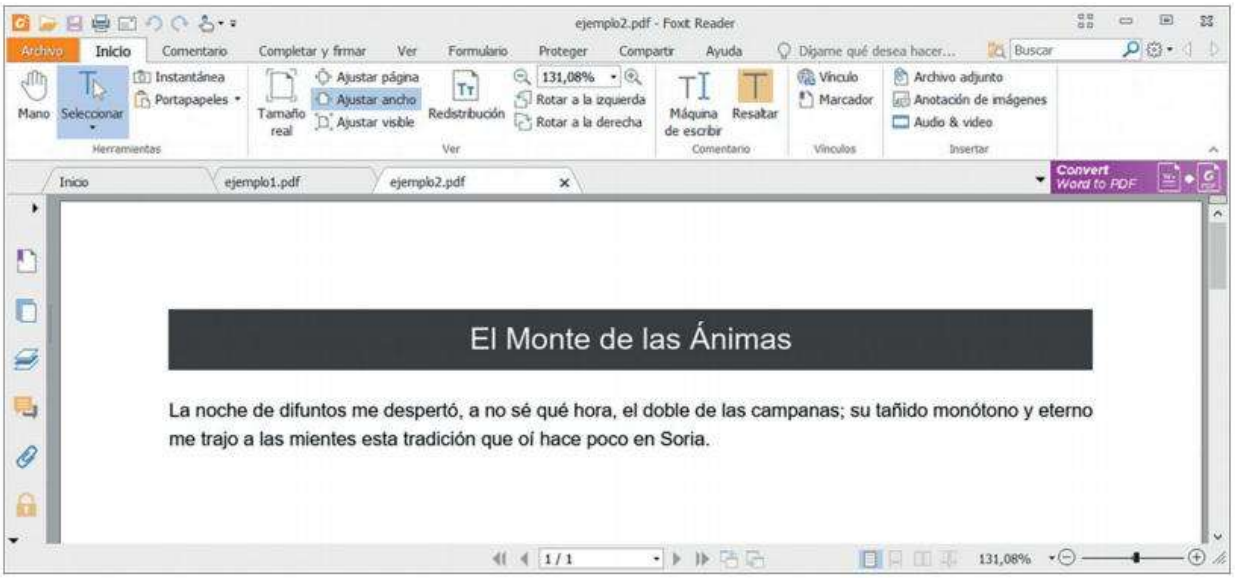


Figura 6.6. *Al obtener los datos de entrada de un documento XML, se incrementa la flexibilidad de XSL-FO.*

6.3.2. Estructura y sintaxis

La estructura básica de un documento XSL-FO es la siguiente:

```
<?xml version="1.0" encoding="utf-8"?>
<fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">
  <fo:layout-master-set>
    <fo:simple-page-master master-name="identificador">
      <!-- CONFIGURACIÓN DE LA PÁGINA -->
    </fo:simple-page-master>
  </fo:layout-master-set>
  <fo:page-sequence master-reference="identificador">
    <!-- CONTENIDO DE LA PÁGINA -->
  </fo:page-sequence>
</fo:root>
```

Además de ser un documento XML bien formado, un fichero XSL-FO incluye los siguientes elementos:

- **<root>.** Elemento principal. Contiene un elemento **<layout-master-set>** y uno o más elementos **<page-sequence>.**
- **<layout-master-set>.** Contiene uno o más elementos <simple-page-master>.
- **<simple-page-master>.** Contiene una configuración de página.
  - **Atributos:**
    - ✓ **master-name.** Nombre de la configuración.
    - ✓ **page-width.** Ancho de la página.

© Ediciones Paraninfo

✓ **page-height.** Alto de la página.

✓ **margin-top.** Margen superior.

✓ **margin-bottom.** Margen inferior.

✓ **margin-left.** Margen izquierdo.

✓ **margin-right.** Margen derecho.

✓ **margin.** Todos los márgenes.

– **Elementos o regiones:**

✓ **<region-body>.** El cuerpo de la página.

✓ **<region-before>.** La cabecera de la página.

✓ **<region-after>.** El pie de la página.

✓ **<region-start>.** El lateral izquierdo de la página.

✓ **<region-end>.** El lateral derecho de la página.

- **<page-sequence>.** Representa un contenedor de objetos de página. Puede ocupar varias páginas del dispositivo si el tamaño lo requiere.

  – **Atributos:**

  ✓ **master-reference.** Referencia una configuración de página.

  – **Elementos:**

  ✓ **<flow>.** Obligatorio. Agrupa el contenido principal de la página.

  ✓ **<static-content>.** Opcional. Puede haber más de un elemento. Agrupa contenido que se repetirá si se ocupa más de una página. Suele utilizarse para los encabezados y pies.

- **<flow> y <static-content>.** Agrupan el contenido de la página.

  – **Atributos:**

  ✓ **flow-name.** Indica la región donde va ubicado el contenido. Admite los valores siguientes:

  • **xsl-region-body.** Cuerpo de la página

  • **xsl-region-before.** Cabecera de la página.

  • **xsl-region-after.** Pie de la página.

  • **xsl-region-start.** Lateral izquierdo de la página.

  • **xsl-region-end.** Lateral derecho de la página.

  – **Elementos:**

  ✓ **<block>.** Agrupa un bloque de contenido.

- **<block>.** Agrupa un bloque de contenido. Se suele utilizar para dar formato a párrafos, títulos, encabezados, figuras y tablas. Admite propiedades para configurar márgenes exteriores, bordes, márgenes interiores, el fondo, la fuente de letra y propiedades referentes al posicionamiento del texto.

© Ediciones Paraninfo

Nota técnica

El listado completo de los atributos del elemento <block> se puede consultar en el siguiente enlace:

https://www.w3.org/TR/xsl/#fo_block

Actividad resuelta 6.2

Creación de tarjeta de presentación en PDF

Crear una tarjeta de presentación en PDF con XSL-FO. En la tarjeta, mostrar el nombre y la profesión de la persona.

Las especificaciones son las siguientes:

- Las dimensiones de la tarjeta son 8,5 cm de ancho por 5,5 cm de alto.
- El texto tiene que estar centrado horizontalmente.
- El tamaño del nombre de la persona debe ser el doble que el de la profesión.
- El color del texto de la profesión debe ser gris.

Solución

```
<?xml version="1.0" encoding="utf-8"?>
<fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">
  <fo:layout-master-set>
    <!-- Tamaño y configuración de la página -->
      <fo:simple-page-master master-name="tarjeta-presentacion" page-
height="5.5cm" page-width="8.5cm" margin-top="2.2cm" margin-bottom="0.5cm">
      <fo:region-body/>
      <fo:region-before extent="0.5cm"/>
      <fo:region-after extent="0.5cm"/>
    </fo:simple-page-master>
  </fo:layout-master-set>
  <!-- Página -->
  <fo:page-sequence master-reference="tarjeta-presentacion">
    <fo:flow flow-name="xsl-region-body">
      <!-- Nombre de la persona. Tamaño de la fuente 16 puntos. -->
      <fo:block font-size="16pt" text-align="center">
      Fernando Paniagua
      </fo:block>
      <!-- Profesión de la persona. Tamaño de la fuente 8 puntos. Color #88888
(gris)-->
      <fo:block font-size="8pt" text-align="center" color="#888888">
      Programador
      </fo:block>
    </fo:flow>
  </fo:page-sequence>
</fo:root>
```

En la Figura 6.7 se puede ver el resultado obtenido.

© Ediciones Paraninfo

Figura 6.7. *Documento PDF generado en la solución de la Actividad resuelta 6.2.*

Actividad resuelta 6.3

Creación de página de libro en PDF

Crear, utilizando XSL-FO, la página de un libro que tenga las siguientes características:

- El tamaño de la página es DIN-A5 (21 cm de alto por 14,85 cm de ancho).
- En el encabezado de la página debe poner «Curso de programación en Java».
- En el pie de la página debe poner «Página» y el número de página.
- En el cuerpo de la página debe mostrarse como título el texto «Bucle for» en color blanco sobre un fondo verde.
- A continuación del título debe mostrarse el enunciado del ejemplo con el texto «El código contenido entre las llaves se ejecuta 100 veces. La variable i toma los valores entre 0 y 99».
- A continuación del enunciado del ejemplo, se debe mostrar el código fuente del ejemplo. Deberá estar correctamente sangrado y en un tipo de letra monoespaciado. El código que mostrar es el siguiente:

```
for (int i=0;i<100;i++){
     //Líneas de código del bucle
}
```

Solución

```
<?xml version="1.0" encoding="utf-8"?>
<fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">
  <fo:layout-master-set>
    <!-- Tamaño y configuración de la página -->
    <fo:simple-page-master master-name="A5" page-height="21cm" page-
width="14.85cm" margin-top="1cm" margin-bottom="1cm" margin-left="2cm" margin-
right="2cm">
      <fo:region-body margin-top="1cm"/>
      <fo:region-before extent="3cm"/>
     <fo:region-after extent="1cm"/>
   </fo:simple-page-master>
 </fo:layout-master-set>
```

© Ediciones Paraninfo

```
  <!-- Página -->
  <fo:page-sequence master-reference="A5">
    <!-- Encabezado -->
    <fo:static-content flow-name="xsl-region-before">
      <fo:block text-align="right">Curso de programación en Java</fo:block>
    </fo:static-content>
    <!-- Pie -->
    <fo:static-content flow-name="xsl-region-after">
      <fo:block text-align="center">
        <!-- Número de página -->
        Página <fo:page-number/>
      </fo:block>
    </fo:static-content>
    <fo:flow flow-name="xsl-region-body">
      <!-- Título de la sección. Color de fondo #55CC55 (verde) -->
      <fo:block font-size="18pt" line-height="30pt" background-color="#55CC55"
color="white" text-align="center" padding-top="2pt">
      Bucle for
      </fo:block>
      <!-- Descripción -->
      <fo:block font-size="10pt" line-height="15pt" text-align="left" margin-
top="20pt">
      El código contenido entre las llaves se ejecuta 100 veces. La variable i
toma los valores entre 0 y 99.
      </fo:block>
      <!-- Código fuente. Utiliza el tipo de letra 'monospace' y respeta los
saltos de línea y los espacios en blanco -->
      <fo:block font-size="10pt" font-family="monospace" linefeed-
treatment="preserve" white-space="pre">
for (int i=0;i&lt;100;i++){
     //Líneas de código del bucle
}
      </fo:block>
    </fo:flow>
  </fo:page-sequence>
</fo:root>
```

En las figuras 6.8 y 6.9 se puede ver el resultado obtenido, mostrándose por separado la parte superior de la página, que incluye el encabezado y el cuerpo, y la parte inferior, que incluye el pie de la página.

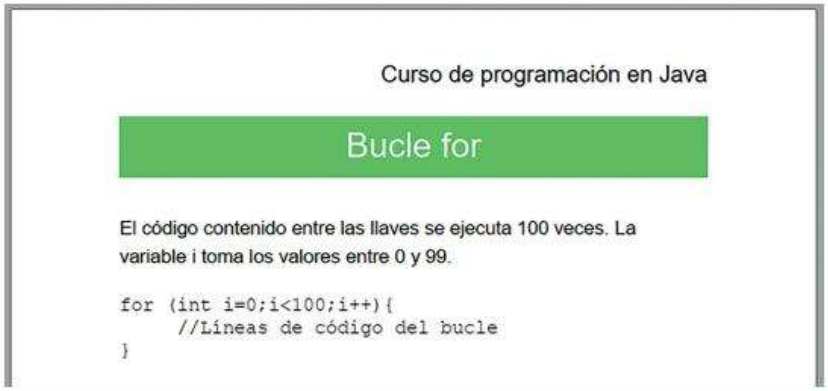


Figura 6.8. *Parte superior del documento PDF generado en la solución de la Actividad resuelta 6.3.*

© Ediciones Paraninfo

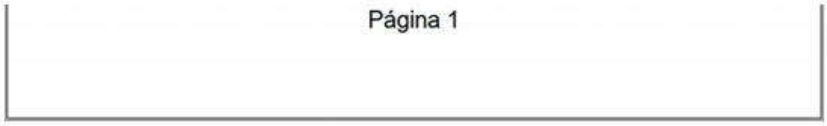


Figura 6.9. *Parte inferior del documento PDF generado en la solución de la Actividad resuelta 6.3.*

Actividad propuesta 6.4

Creación de una invitación de boda en PDF

Utilizando XSL-FO, crea un documento PDF con la invitación a una boda.

El tamaño del documento será de 20 cm de ancho por 9 cm de alto.

Los datos que debe mostrar son:

- Nombre de los novios (por ejemplo, «Julieta & Romeo»).
- Texto de la invitación (por ejemplo, «Nos hace mucha ilusión compartir con vosotros este día tan especial»).
- Fecha y hora de la ceremonia.
- Lugar de la ceremonia.
- Lugar de la celebración.
- Teléfono para confirmar la asistencia.

Puedes inspirarte en invitaciones de boda que encuentres en la web.

6.4. Herramientas de transformación XML

Para realizar la transformación de un documento XML utilizando XSL se necesita una herramienta que realice el proceso. Existen múltiples alternativas, desde aplicaciones de escritorio hasta sistemas en línea accesibles a través de la web, además de la posibilidad de programar un procesador propio en algún lenguaje de programación moderno.

6.4.1. Navegadores web

Los navegadores web más populares incorporan la capacidad de realizar transformaciones XSLT.

Las transformaciones se realizan simplemente abriendo el documento XML que contiene la referencia al documento XSL.

Por razones de seguridad, el acceso al documento XML y, por lo tanto, al documento XSL, deben realizarse a través de un servidor web mediante el protocolo HTTP. Si se quiere evitar esta restricción, es necesario modificar los parámetros de seguridad que impiden las referencias entre ficheros locales.

© Ediciones Paraninfo

6.4.2. XMLPad

Este editor proporciona una forma sencilla y versátil de aplicar transformaciones XSLT. Permite seleccionar diversos tipos de salida, el programa con el que se quiere abrir el fichero obtenido e incluso seleccionar el procesador que se va a utilizar.

Lo pasos para realizar una transformación son los siguientes:

- Abrir el documento XML.
- Crear un escenario de transformación (opción «Tools/Transforms» del menú).

Figura 6.10. *Muchos editores de escritorio realizan transformaciones XSLT.*

- Ejecutar la transformación (opción «XML/Apply Transform» del menú), seleccionando el escenario correspondiente si se solicita.

El editor muestra en la pestaña «Preview» el resultado de la transformación.

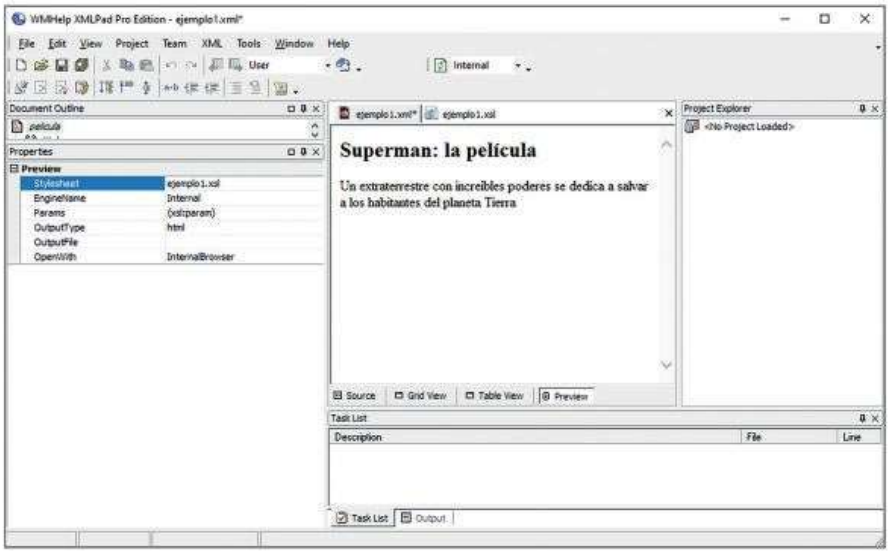


Figura 6.11. *XMLPad permite seleccionar la herramienta de presentación de los resultados de las transformaciones.*

© Ediciones Paraninfo

6.4.3. XML Notepad

XML Notepad también incorpora la capacidad de realizar transformaciones XSLT.

En este editor el proceso de transformación es tan sencillo como abrir el documento XML y seleccionar la pestaña XSL Output para consultar el resultado. No admite ningún tipo de configuración, pero es una interesante herramienta para tomar contacto con las transformaciones XLST.

Figura 6.12. *Los editores son muy útiles para probar las transformaciones XSLT.*

6.4.4. Editores en línea

Existen múltiples sitios web que proporcionan la capacidad de realizar transformaciones XSLT en línea, ya sea copiando el código XML y XSLT a un formulario disponible en la página web o subiendo los ficheros que contienen dicha información.

Las transformaciones se realizan generando el código de resultado que podrá descargarse para utilizarse como se desee.

Algunos de los múltiples editores online existentes son *Free Online XSL Transformer,* de la web **Freeformatter.com**; *Free Online XSLT Test Tool,* de la web **xsltest.appspot.com**, o *XSLT Tryit Editor,* de la web **w3schools.com** del cual se recoge una captura de pantalla en la Figura 6.13.

Recuerda

Las herramientas en línea o herramientas web son muy cómodas y útiles para realizar pruebas y prototipos, pero en ocasiones la funcionalidad que proporcionan es insuficiente.

© Ediciones Paraninfo

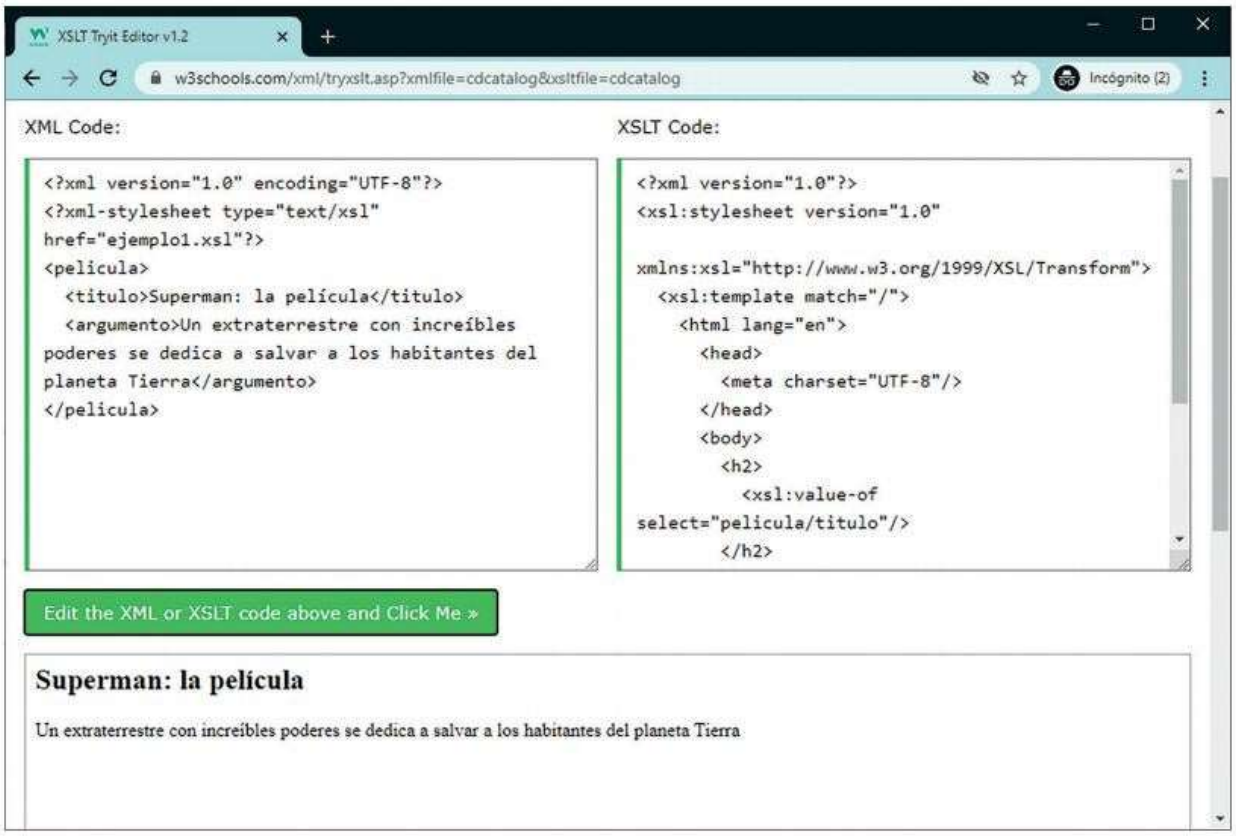


Figura 6.13. *Los editores en línea son más agiles, ya que no requieren instalación.*

6.4.5. Procesadores a través de programa

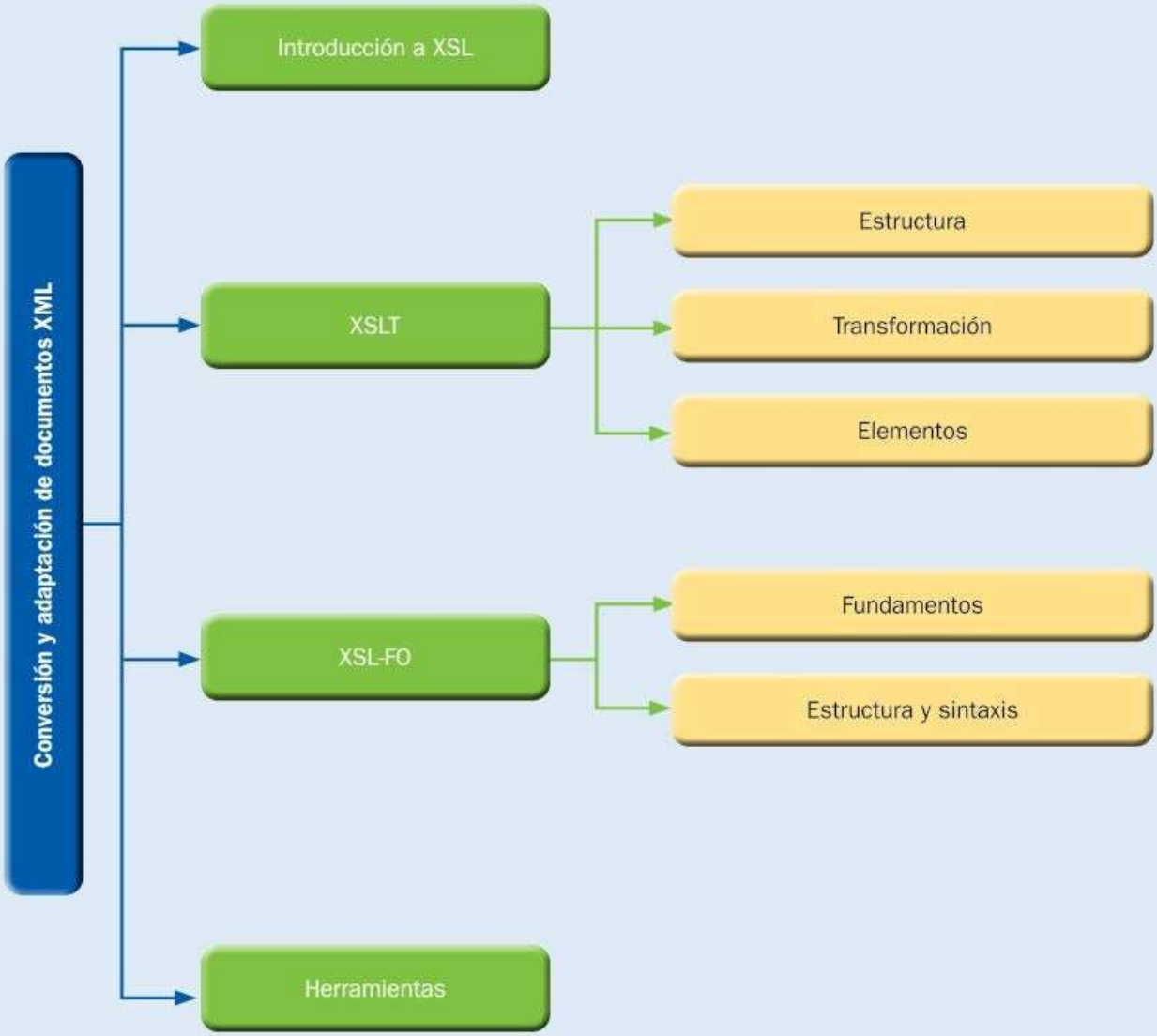
Todas las transformaciones que se presentan en esta unidad se pueden realizar desde programa e integrar en cualquier aplicación.

Las transformaciones XSL-FO podrían utilizarse, por ejemplo, para generar las facturas en formato PDF a partir de datos de los clientes almacenados en documentos XML para archivarse o enviarse por correo electrónico. Apache FOP dispone de un API para realizar este tipo de acciones desde aplicaciones Java.

De igual forma, es posible realizar transformaciones con XSLT de documentos XML desde programa utilizando las librerías adecuadas.

© Ediciones Paraninfo

MAPA CONCEPTUAL



© Ediciones Paraninfo

Actividades de comprobación

6.1. ¿Qué significan las siglas XSL?

a) *eXtensible Stylized Language.*
b) *eXtensible Stylesheet Language.*
c) *eXtra Stylesheet Language.*
d) *eXtra Stylized Language.*

6.2. ¿Cuál de las siguientes siglas corresponde a tecnologías XSL?

a) XSLT.
b) SLTX.
c) TLSX.
d) STXL.

6.3. ¿Cuál de las siguientes opciones no corresponde a tecnologías XSL?

a) XLS-FO.
b) XSLT.
c) XQuery.
d) XPath.

6.4. ¿Cuál es el elemento de XLST que se utiliza para mostrar el contenido de un elemento XML?

a) <content-of>
b) <text-of>
c) <value-of>
d) <index-of>

6.5. ¿Cuál es el elemento de XLST que se emplea para recorrer un conjunto de elementos XML?

a) <while-true>
b) <sort-of>
c) <for-each>
d) <do-while>

6.6. ¿Cuál es el elemento de XLST que se utiliza para incluir comentarios?

a) <tag>
b) <free-text>
c) <comment>
d) <note>

6.7. ¿Qué papel tiene XSL-FO en la familia XSL?

a) Hacer transformaciones.
b) Configurar formatos de salida.
c) Hacer consultas.
d) Todas las respuestas son correctas.

6.8. ¿Qué tipo de formato se puede generar con XSL-FO?

a) PDF.
b) PNG.

© Ediciones Paraninfo

c) RTF.
d) Todas las respuestas son correctas.

6.9. ¿Cúal es el elemento principal de XSL-FO?
a) <main>
b) <node>
c) <root>
d) <layer>

6.10. ¿En qué elemento de XSL-FO se suelen incluir los encabezados de página?
a) <flow>
b) <static-content>
c) <region-start>
d) <block>

6.11. ¿Qué elemento indica en XSL-FO que comienza una página?
a) <region-body>
b) <page-sequence>
c) <simple-page-master>
d) <layout-master-set>

6.12. ¿Cómo se puede utilizar XSL-FO?
a) Desde un editor en línea.
b) Desde un editor de escritorio.
c) Desde un programa a través de librerías.
d) Todas las respuestas son correctas.

Actividades de aplicación

6.13. Explica qué función tiene XSLT dentro de la familia XSL.

6.14. ¿Qué función tiene XPath en XSLT?

6.15. Desarrolla qué función tiene XSL-FO dentro de la familia XSL.

6.16. ¿Qué función tiene XPath en XSL-FO?

6.17. Explica qué relación tienen XSTL y XSL-FO entre sí dentro de la familia XSL.

6.18. En el portal de la Agencia Estatal de Meteorología, se publican en XML los pronósticos por municipios (http://www.aemet.es/es/eltiempo/prediccion/municipios). Analiza los datos proporcionados en dicho XML y diseña una transformación en XSLT para presentar los datos más relevantes.

6.19. En el portal The Open Movie Database (http://www.omdbapi.com/), se proporcionan servicios web con información sobre películas. Permite, mediante un formulario, hacer consultas de prueba y acceder a los datos proporcionados en los formatos XML y JSON. Obtén los datos de una película, crea un documento XML con ellos y haz una presentación en HTML utilizando XSLT.

© Ediciones Paraninfo

6.20. Disponiendo de la estructura de un hotel (relación de habitaciones incluyendo su capacidad, número de camas y si disponen o no de aire acondicionado), diseña una transformación con XSLT para obtener un nuevo documento XML en la que solo aparezcan las habitaciones con capacidad para más de cuatro personas.

6.21. Comprueba alguna de tus soluciones de las actividades de aplicación en tres editores en línea distintos. Compara los resultados y determina qué editor te parece mejor justificando la respuesta.

6.22. A partir de un documento XML diseñado por ti, crea mediante XSL-FO un documento PostScript. Busca en la web un visor en línea de este tipo de formatos y comprueba que se ha generado correctamente.

6.23. A partir de un documento XML diseñado por ti, crea mediante XSL-FO un documento PNG. Comprueba con cualquier editor gráfico que se ha generado correctamente.

6.24. A partir de un documento XML diseñado por ti, crea mediante XSL-FO un documento PDF en el que se muestre el número de página. El documento debe ocupar más de una página. Comprueba que todas las páginas se numeran correctamente.

Actividades de ampliación

6.25. Busca la lista completa de elementos XSLT. Selecciona dos que te parezcan importantes y que no se traten en esta unidad. Pruébalos.

6.26. Profundiza en XSL-FO. Averigua cómo se crean tablas.

6.27. Busca más información sobre en XSL-FO, y descubre cómo se crean listas.

6.28. ¿Se pueden realizar transformaciones XSLT en los lenguajes de programación: C#, PHP, C y Python?

6.29. Busca información referente a cómo se utiliza Apache FOP desde una aplicación Java.

6.30. JSON se presenta en muchas ocasiones como una alternativa a XML, pero ¿dispone de las mismas herramientas? Determina si existen tecnologías equivalentes a XSLT y XSL-FO para el lenguaje JSON.

Enlaces web de interés

- **XSL en W3C -** https://www.w3.org/Style/XSL/
  (Información sobre la familia XSL en la web del W3C)

© Ediciones Paraninfo

- **XSLT en MDN -** https://developer.mozilla.org/es/docs/Web/XSLT
  (Información sobre XSLT en la web para desarrolladores de Mozilla)

- **Apache FOP -** https://xmlgraphics.apache.org/fop/
  (Procesador de XSL-FO)

- **PassiveText -** http://projects.oucs.ox.ac.uk/passivetex/
  (Librería de XSL-FO)

- **XSLT en Java -** https://docs.oracle.com/javase/tutorial/jaxp/xslt/transformingXML.html
  (Información sobre transformaciones desde Java en la web de Oracle)

- **Editor XMLPAD -** https://xmlpad-mobile.com/
  (Editor XML con XSLT)

- **Editor XmlNotepad -** https://github.com/microsoft/XmlNotepad
  (Editor XML con XSLT)

- **Editor XML Freeformater.com -** https://www.freeformatter.com/xsl-transformer.html
  (Editor en línea XML con XSLT)

- **Online XSLT Test Tool -** https://xslttest.appspot.com/
  (Editor en línea XML con XSLT)

- **Editor de pruebas de w3school -** https://www.w3schools.com/xml/tryxslt.asp?xmlfile=cdcatalog&xsltfile=cdcatalog
  (Editor en línea XML con XSLT)

© Ediciones Paraninfo

UNIDAD 7

JSON, el lenguaje ligero. Creación y utilización

Objetivos

- Exponer las características principales del formato JSON.
- Detallar la estructura y sintaxis de este lenguaje.
- Presentar algunos de los usos más habituales que se da a este tipo de ficheros.
- Descubrir algunas herramientas para el manejo de los ficheros JSON.

Contenidos

- 7.1. Introducción a JSON
- 7.2. Estructura y sintaxis del formato JSON
- 7.3. Aplicaciones de JSON
- 7.4. Herramientas de edición y consulta de documentos JSON

Introducción

El término «lenguaje de marcas» engloba un amplio conjunto de lenguajes. Cada uno de ellos tiene sus propias reglas y características. JSON es diferente, principalmente, por su sencillez. No tiene etiquetas y está definido por una pequeña y compacta sintaxis capaz de representar estructuras de datos jerárquicas utilizando un reducido conjunto de tipos de datos.

Gracias a estas características, JSON es un lenguaje ampliamente utilizado en múltiples entornos: ficheros de configuración, intercambio de datos entre aplicaciones y, especialmente, en el desarrollo de aplicaciones web.

7.1. Introducción a JSON

JSON es el acrónimo de *JavaScript Object Notation* y es un lenguaje de marcado muy sencillo apropiado, sobre todo, para el intercambio de datos.

JSON es el formato en el que JavaScript representa internamente sus objetos. A pesar de ello, su utilización está tan extendida que se considera un lenguaje independiente.

La sintaxis de JSON es muy simple y los documentos son más ligeros que en otros lenguajes como XML, siendo estas dos características algunas de las razones de su éxito. Los documentos escritos en este lenguaje son extremadamente sencillos de procesar desde JavaScript, ya que es su lenguaje de programación natural. También resulta muy sencillo procesarlos en el resto de los lenguajes de programación en los que hay disponibles procesadores, que son prácticamente todos los lenguajes de programación modernos.

Aparte de la validación sintáctica, JSON no dispone de forma nativa de mecanismos de validación análogos a los existentes en XML como DTD o XML Schema, aunque hay algunos proyectos desarrollados en paralelo que proporcionan funcionalidades similares.

En el siguiente código JavaScript se muestra la conversión de una cadena de caracteres a objeto JSON (mediante el método **parse** de la clase **JSON**) y el acceso a sus componentes mediante la simple referencia al nombre de los parámetros (película y director en el ejemplo). Como se puede comprobar, el acceso a los datos contenidos en el objeto JSON es muy sencillo.

```
let cadenaJSON = '{"pelicula": "El Resplandor","director": "Stanley
Kubrick"}';
let objetoJSON = JSON.parse(cadenaJSON);
let tituloPelicula = objetoJSON.pelicula;
let directorPelicula = objetoJSON.director;
```

Además de utilizarse como lenguaje de intercambio entre sistemas, JSON (o alguno de sus dialectos) se emplea para almacenar información en bases de datos no relacionales. Los mismos datos expresados en JSON suelen ocupar menos espacio que en otros

© Ediciones Paraninfo

lenguajes, lo que lo convierte en una solución más eficiente para el almacenamiento de grandes volúmenes de información.

En los dos siguientes bloques de código se muestran los mismos datos representados en los lenguajes XML y JSON. Como se puede observar, el documento JSON ocupa menos tamaño (274 bytes) que la versión XML (341 bytes), alrededor de un 20 % menos de espacio.

Documento XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<consolas>
    <consola>
        <nombre>XBox Series</nombre>
        <fabricante>Microsoft</fabricante>
        <anyo>2020</anyo>
    </consola>
    <consola>
        <nombre>Play Station 5</nombre>
        <fabricante>Sony</fabricante>
        <anyo>2020</anyo>
    </consola>
</consolas>
```

Documento JSON:

```
{
    "consolas":[
        {
            "nombre":"XBox Series",
            "fabricante":"Microsoft",
            "anyo":2020
        },
        {
            "nombre":"Play Station 5",
            "fabricante":"Sony",
            "anyo":2020
        }
    ]
}
```

En ocasiones se suscita un debate sobre si JSON es mejor o no que XML. Dicho debate no debería plantearse de forma general. Aunque ambos lenguajes se suelen utilizar a menudo para resolver problemas similares, las capacidades de ambos son muy diferentes. XML dispone de una sintaxis muy completa y de muchas herramientas y tecnologías alrededor de las que carece JSON. En entornos en los que la fiabilidad de los datos es un factor relevante, XML suele ser la mejor alternativa, debido, entre otras razones, a la existencia de mecanismos de validación, con las implicaciones que estos tienen sobre las

© Ediciones Paraninfo

relaciones, los tipos de datos o las cardinalidades de los elementos. Por otra parte, en entornos en los que la robustez de los datos esté garantizada de partida o no sea determinante, JSON puede ser una opción más eficiente. En entornos en los que el volumen de transferencia de datos sea muy alto, JSON será la opción ideal, siempre que la fiabilidad de los datos no esté comprometida o no sea un factor crítico. En resumen, no existe un lenguaje mejor o peor, sino uno más adecuado para cada uso concreto.

Figura 7.1. *XML y JSON son lenguajes complementarios, no competencia.*

7.2. Estructura y sintaxis del formato JSON

Tanto la estructura como la sintaxis de JSON son muy simples. Los ficheros se almacenan con la extensión **json.** Debido a su sencillez, este lenguaje no permite incluir comentarios.

JSON dispone de dos tipos de elementos:

- Objetos.
- Listas o arrays.

Al tratarse de estructuras de datos tan simples, son universales y, por lo tanto, están disponibles en prácticamente todos los lenguajes de programación. Gracias a esto, JSON está tan extendido.

La sintaxis de los objetos es la siguiente:

- Están delimitados por llaves.
- Contienen una lista de pares de clave-valor separados por comas.
- Los nombres de las claves se separan de los valores por dos puntos.
- Los nombres de las claves se escriben entre comillas dobles.
- Los **valores** pueden ser:
  - Cadenas de caracteres. Se escriben entre comillas dobles.
  - Números, aceptando enteros y decimales.
  - Valores lógicos (**true** y **false**).

© Ediciones Paraninfo

– El valor **null.**

– Objetos.

– Arrays de objetos.

En la Tabla 7.1 se muestran algunos ejemplos de diversos tipos de objeto.

Tabla 7.1. Posibles combinaciones de pares tipo-valor en JSON

Tipo	Ejemplo
Un único par tipo-valor, siendo el valor una cadena de caracteres.	`{"titulo":"Los Ojos de Julia"}`
Un único par tipo-valor, siendo el valor un número entero.	`{"anyo":2010}`
Un único par tipo-valor, siendo el valor un número decimal.	`{"calificacion":6.1}`
Un único par tipo-valor, siendo el valor de tipo lógico.	`{"estreno-cine":true}`
Dos pares tipo-valor.	`{ "titulo":"Los Ojos de Julia", "anyo":2010 }`
Tres pares tipo-valor, siendo el último de ellos un objeto.	`{ "titulo":"Los Ojos de Julia", "anyo":2010, "director":{ "nombre":"Guillem Morales", "nacionalidad":"Español" } }`

Las **listas** o **arrays** son colecciones de elementos, que pueden a su vez ser objetos o arrays.

La sintaxis de las listas es la siguiente:

- Están delimitadas por corchetes.
- Los elementos están separados por comas.

En su versión más sencilla, las listas pueden contener colecciones de valores simples:

```
{
    "titulo" : "Nosferatu, vampiro de la noche",
    "titulo-original" : "Nosferatu: Phantom der Nacht",
    "director" : "Werner Herzog",
    "reparto" : ["Klaus Kinski","Bruno Ganz","Jacques Dufilho"]
}
```

© Ediciones Paraninfo

En su versión más elaborada, las listas pueden contener colecciones de objetos:

```
{
    "titulo":"Los Ojos de Julia",
    "anyo":2010,
    "director":{
        "nombre":"Guillem Morales",
        "nacionalidad":"Español"
    },
    "reparto":[
        {
            "nombre":"Belén Rueda",
            "personaje":"Julia"
        },
        {
            "nombre":"Lluis Homar",
            "personaje":"Isaac"
        },
        {
            "nombre":"Francesc Orella",
            "personaje":"Inspector Dimas"
        }
    ]
}
```

Un documento JSON se forma con un objeto que contiene objetos o listas de objetos, creando una estructura jerárquica en forma de árbol.

Actividad resuelta 7.1

Representación de un lenguaje de programación

Representar en un fichero JSON un lenguaje de programación. La información que va a contener es la siguiente:

- Nombre del lenguaje.
- Año de creación.
- Nombres de los autores.
- Indicador de si es compilado o no.

Solución

```
{
    "nombre":"C",
    "anyo-presentacion":1972,
    "creadores":"Dennis Ritchie",
    "compilado":true
}
```


© Ediciones Paraninfo

Actividad resuelta 7.2

Representación de un equipo de baloncesto

Representar en un fichero JSON un equipo de baloncesto. La información que va a contener es la siguiente:

- Nombre del equipo.
- Color de la equipación.
- Nombre del estadio.
- Nombre del entrenador.
- Indicador de si la plantilla del equipo se corresponde con una plantilla histórica.
- Referencia a su página web (si está disponible).
- Jugadores. Por cada jugador se almacena el nombre y el dorsal. Se agrupan según las siguientes posiciones.
- Base.
- Alero.
- Pívot.

Solución

```
{
    "nombre":"Los Angeles Lakers",
    "colores":"Púrpura y Oro",
    "estadio":"Staples Center",
    "entrenador":"Phil Jackson",
    "historico":true,
    "web":null,
    "bases":[
        {
            "nombre":"Magic Johnson",
            "dorsal":32
        },
        {
            "nombre":"Jerry West",
            "dorsal":44
        }
    ],
    "aleros":[
        {
            "nombre":"James Worthy",
            "dorsal":42
        },
        {
            "nombre":"Kobe Bryant",
            "dorsal":8
        }
    ],
   "pivotes":[
        {
            "nombre":"Kareem Abdul-Jabbar",
            "dorsal":33
```

© Ediciones Paraninfo

```
        },
        {
            "nombre":"Wilt Chamberlain",
            "dorsal":13
        },
        {
            "nombre":"Pau Gasol",
            "dorsal":16
        }
    ]
}
```

Actividades propuestas

7.1. Representación de una obra pictórica

Crea un documento JSON que represente un cuadro. La información que debe contener es la siguiente:

- Título del cuadro.
- Nombre del autor.
- Año de creación.
- Indicador de si está expuesta en un museo.
- Nombre del museo. Puede ser nulo.

7.2. Representación de la obra de un pintor

Crea un documento JSON que represente la obra de un pintor. La información que debe contener es la siguiente:

- Nombre del pintor.
- Fecha de nacimiento.
- Lugar de nacimiento.
- Fecha de defunción. Puede ser nula.
- Lugar de defunción. Puede ser nula.
- Listado de obras. Por cada obra dispone de la siguiente información:
  - Título del cuadro.
  - Año de creación.
  - URL a una fotografía del cuadro alojada en internet.

7.3. Detección de errores

El siguiente documento JSON contiene **cuatro** errores de sintaxis. Intenta encontrarlos y solucionarlos sin utilizar el ordenador.

```
{
    "pelicula": "El Resplandor",
    "director": "Stanley Kubrick"
    "estreno"; 1980,
    "reparto": [
```

© Ediciones Paraninfo

```
        {
            "nombre": "Jack Nicholson",
            "personaje": "Jack Torrance"
        },
        {
            "nombre": "Shelley Duvall",
            "personaje": "Wendy Torrance"
        },
        {
            "nombre": "Danny Lloid",
            "personaje": "Danny Torrance"
    ]
}
```

7.3. Aplicaciones de JSON

Pese a su simplicidad, el lenguaje JSON se utiliza en varios contextos distintos. Esto es posible gracias a que, como ocurre con los demás lenguajes de marcas, está formado por texto plano. Además, JSON utiliza tipos de datos muy básicos y universales, facilitando la existencia de procesadores en cualquier lenguaje de programación.

7.3.1. Configuración de aplicaciones

Uno de los usos más inmediatos de los ficheros JSON consiste en dar soporte al mantenimiento de la configuración de aplicaciones.

Cuando se desarrolla un sistema informático, algunos de los problemas habituales que hay que resolver son los referentes a la gestión de la configuración y al almacenamiento del estado de la aplicación. JSON es un lenguaje perfecto para dar soporte a estas funcionalidades.

Se puede ilustrar el encaje de JSON en este tipo de aplicación con el siguiente ejemplo.

Ejemplo:

Prácticamente todas las aplicaciones permiten crear configuraciones personalizadas para los usuarios que las utilizan. Estos pueden establecer los valores apropiados para los parámetros correspondientes y adaptar los sistemas a sus gustos o necesidades.

Supongamos una aplicación que permite configurar los siguientes parámetros:

- Color de fondo.
- Tamaño del texto.
- Color del texto.
- Idioma.
- Pantalla de inicio.

© Ediciones Paraninfo

Una posible estructura de fichero JSON para soportar esta configuración sería la siguiente:

```
{
    "usuarios" : [
        {
            "identificador":"jhipolito",
            "configuracion": {
                    "color-fondo":"#AA99AA",
                    "tamaño-texto":12,
                    "color-texto":"#441238",
                    "idioma":"es",
                    "pantalla-inicio":"listado-clientes"
            }
        },
        {
            "identificador":"alanchas",
            "configuracion": {
                    "color-fondo":"#8911F8",
                    "tamaño-texto":14,
                    "color-texto":"#311103",
                    "idioma":"it",
                    "pantalla-inicio":"listado-deudores"
            }
        }
    ]
}
```

7.3.2. Intercambio de datos y servicios web

Utilizar este lenguaje para representar los datos que intercambian aplicaciones entre sí es, probablemente, el uso más habitual que se le da a los ficheros JSON.

Desde el punto de vista de los sistemas de información, las empresas suelen tener múltiples aplicaciones informáticas para dar soporte a su actividad cotidiana: gestión de pedidos, almacén, facturación, clientes, contabilidad, etc. Estos sistemas pueden estar programados en lenguajes distintos y ejecutarse en diferentes ordenadores. A pesar de ello, las aplicaciones no son estancas entre sí, sino que tienen que compartir datos tanto en diferido como en tiempo real.

El intercambio de datos se puede realizar de varias maneras: a través de bases de datos o de ficheros alojados en ubicaciones compartidas. En muchos de estos casos, JSON puede utilizarse como formato para representar los datos, garantizando la interoperatividad: todos los lenguajes y todos los sistemas operativos son capaces de procesar ficheros de texto.

Por otra parte, las comunicaciones en tiempo real entre sistemas requieren formatos de datos compatibles y representaciones susceptibles de ser enviadas por procedimientos telemáticos como *sockets,* RPC o servicios web.

© Ediciones Paraninfo

Nota técnica

Los *sockets,* RPC *(Remote Procedure Call* o «llamada a procedimiento remoto») y los servicios web son mecanismos de comunicación entre sistemas. Tienen un nivel de abstracción distinto, perteneciendo los *sockets* al más bajo y los servicios web al más alto. Todos ellos admiten, por supuesto, el paso de texto plano, por lo que pueden intercambiar datos expresados en JSON sin dificultad.

Hay que considerar que el mayor flujo de datos en computación se produce entre los navegadores y los servidores web. Por un lado, el navegador, actuando como cliente, ejecuta un programa escrito en lenguaje JavaScript que procesa JSON de forma nativa; en el otro extremo, actuando como servidor, un programa escrito en Java, C, Python o PHP, que dispone de librerías para procesar los documentos JSON de forma sencilla.

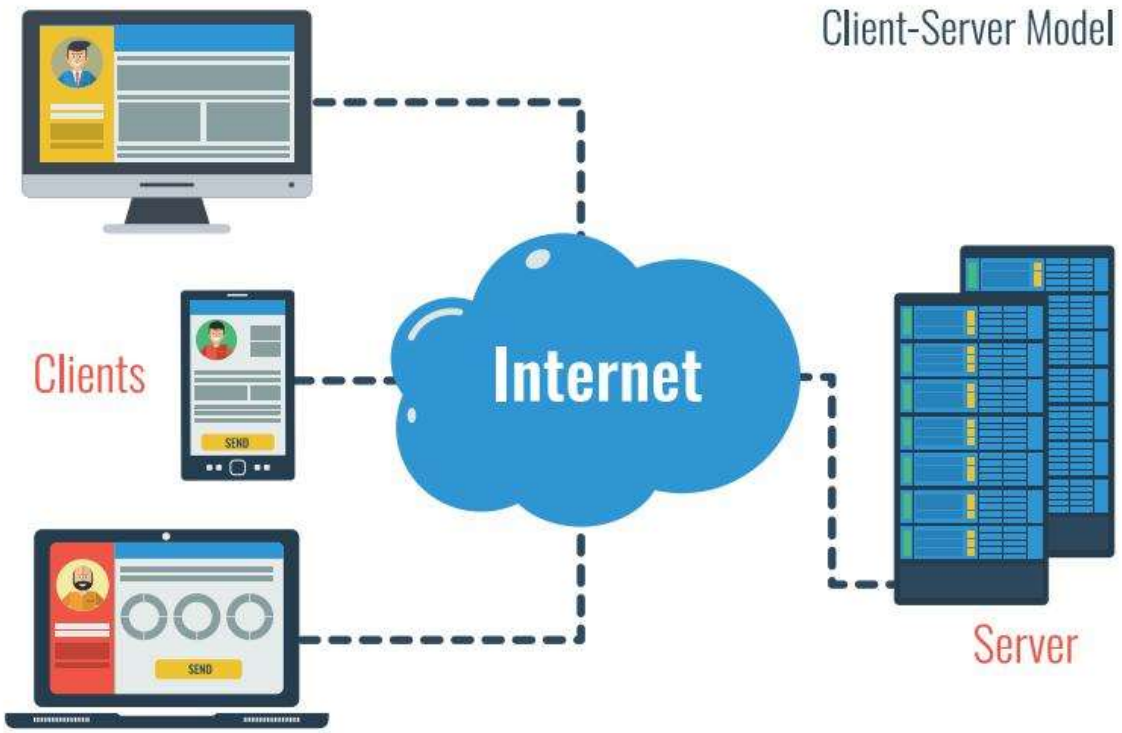


Figura 7.2. *Muchas arquitecturas de aplicaciones web modernas utilizan JSON.*

En las aplicaciones web tradicionales, el cliente (navegador web) realiza una petición HTTP al servidor y este devuelve un documento HTML que interpreta el navegador. Los datos de la llamada están embebidos en la petición HTTP y los datos de la respuesta están incluidos en un documento HTML orientado a la presentación, lo que dificulta su procesamiento.

Desde hace un tiempo se ha incorporado otra forma de hacer las aplicaciones web, mediante el desarrollo de unos componentes llamados **servicios web** o ***Web Services.*** La transferencia de datos se sigue realizando mediante el protocolo HTTP, pero, en lugar de intercambiar peticiones y código HTML, intercambian datos expresados en JSON. En este tipo de aplicación, el cliente (ya no tiene por qué ser un navegador, puede ser un programa escrito en cualquier lenguaje) realiza llamadas al servidor enviando y recibiendo datos en

© Ediciones Paraninfo

formato JSON. Con los datos recibidos, el programa cliente podrá construir la representación de los datos en un navegador o en una aplicación móvil o de escritorio. Además, al tratarse de datos fácilmente procesables, la aplicación cliente podrá procesarlos sin dificultad para realizar acciones adicionales a la presentación de estos.

Nota técnica

HTTP es el protocolo sobre el que se realizan las comunicaciones entre máquinas a través de la web.

Al programa alojado en el servidor que, mediante el protocolo HTTP, recibe peticiones, realiza un proceso y devuelve datos se le denomina **servicio web.** El formato de los datos que recibe y devuelve un servicio web lo decide el diseñador de este, pero habitualmente se utilizan XML y JSON.

Por ejemplo, la web OMDb API proporciona un servicio web con información sobre películas. Esta web ofrece la posibilidad de obtener la respuesta eligiendo que esta se represente mediante JSON o XML.

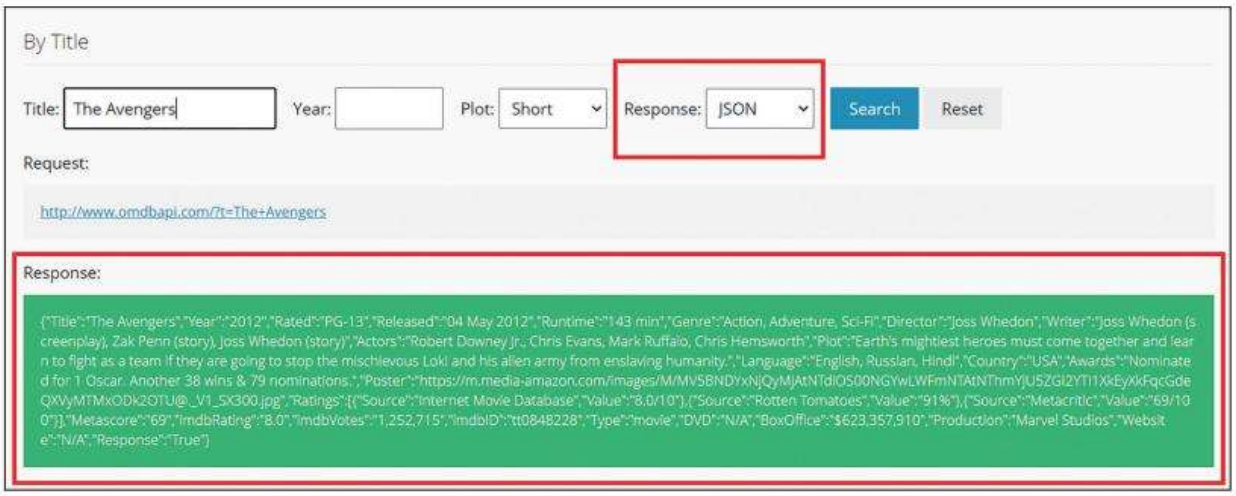


Figura 7.3. *Muchos servicios web proporcionan los datos en formato JSON.*

Los servicios web ofrecen habitualmente JSON como opción para representar los datos intercambiados entre cliente y servidor debido a sus características, teniendo gran aceptación por parte de diseñadores y programadores.

7.3.3. Almacenamiento

JSON se utiliza como formato para almacenar datos en algunos sistemas.

Por ejemplo, la base de datos NoSQL MongoDB utiliza un dialecto de JSON llamado BSON (es una versión binaria de JSON).

Otro ejemplo es la base de datos Firebase Realtime Database de Google. Es una base de datos multiplataforma en tiempo real alojada en la nube. Los datos se almacenan en

© Ediciones Paraninfo

formato JSON, lo que facilita el tratamiento desde los programas clientes de la base de datos.

En la Figura 7.4 se muestra la interfaz gráfica desde la que se pueden editar y consultar los datos almacenados en la base de datos.

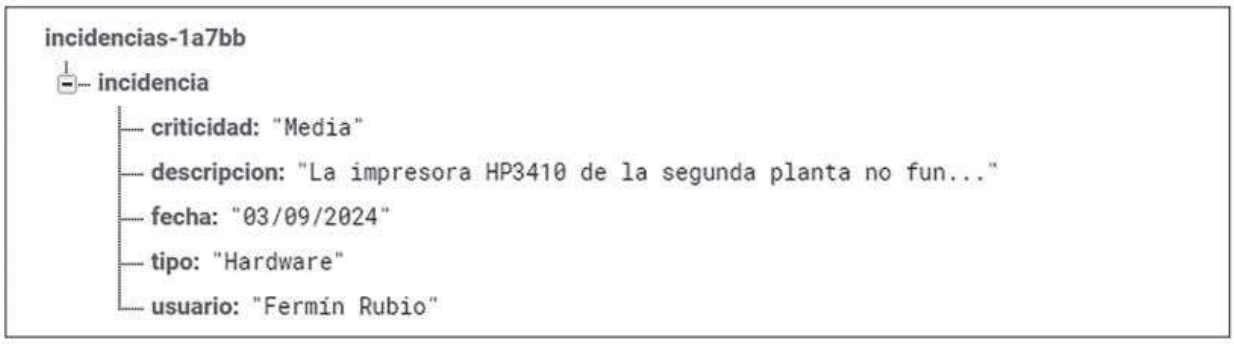


Figura 7.4. *Un documento JSON refleja una estructura en forma de árbol.*

Firebase Realtime Database proporciona la posibilidad de importar y exportar ficheros JSON, para inicializar la base de datos o para recuperar los datos almacenados en esta.

Figura 7.5. *Debido a sus características, JSON se utiliza en sistemas de bases de datos no relacionales.*

La exportación genera una copia en formato JSON de los datos:

```
{
  "incidencia" : {
    "criticidad" : "Media",
    "descripcion" : "La impresora HP3410 de la segunda planta no funciona",
    "fecha" : "03/09/2024",
    "tipo" : "Hardware",
    "usuario" : "Fermín Rubio"
  }
}
```

MongoDB y Firebase Realtime Database son dos ejemplos representativos de cómo JSON se utiliza en sistemas de almacenamiento de información.

© Ediciones Paraninfo

7.4. Herramientas de edición y consulta de documentos JSON

JSON se representa como una cadena de caracteres, por lo que, al igual que ocurre con el resto de los lenguajes de marcas, cualquier editor permite consultar y editar el contenido. El Bloc de Notas de Windows, Gnote en Linux o TextEdit en macOS son ejemplos de editores básicos en distintos sistemas operativos capaces de permitir la consulta y edición de documentos JSON.

No obstante, al igual que ocurre con el resto de los lenguajes de marcas, es recomendable disponer de un editor más completo, que realice validación sintáctica y que presente el documento de forma gráfica para que la consulta sea más sencilla.

Los editores pueden ser **editores de escritorio, editores en línea** o **navegadores.**

7.4.1. Editores de escritorio

Los editores de escritorio pueden no estar diseñados de manera específica para tratar documentos escritos con lenguajes de marcas, pero la mayoría dispone de soporte para JSON de forma nativa o mediante componentes añadidos.

Por ejemplo, el editor Visual Studio Code incorpora un procesador de JSON que verifica la sintaxis de los documentos y los representa según un código de colores.

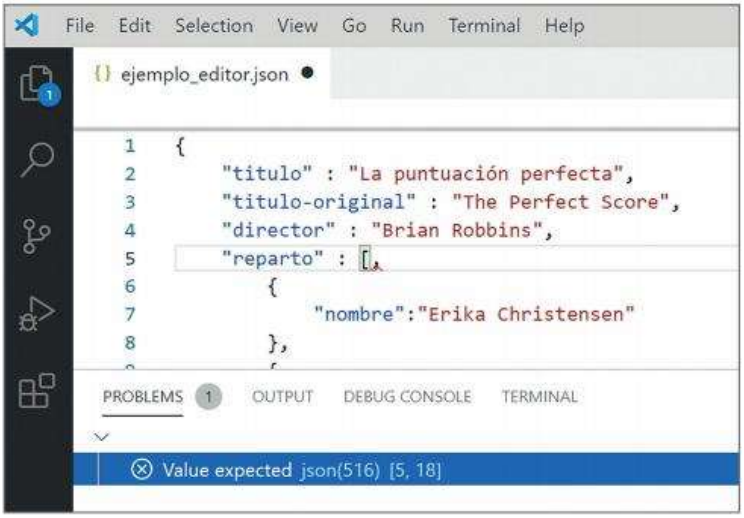


Figura 7.6. *Cualquier editor sirve para escribir un fichero JSON.*

7.4.2. Editores en línea

Los editores en línea son una interesante opción para consultar y editar documentos JSON. No requieren instalación, ya que se utilizan a través del navegador. Son muy completos y existen muchas alternativas gratuitas.

© Ediciones Paraninfo

Por ejemplo, JSON Editor Online (https://jsoneditoronline.org/) permite abrir documentos desde el sistema de archivos del ordenador o desde una URL, valida la sintaxis, muestra los elementos con un código de colores y en forma de árbol, y es capaz de comparar documentos y encontrar diferencias.

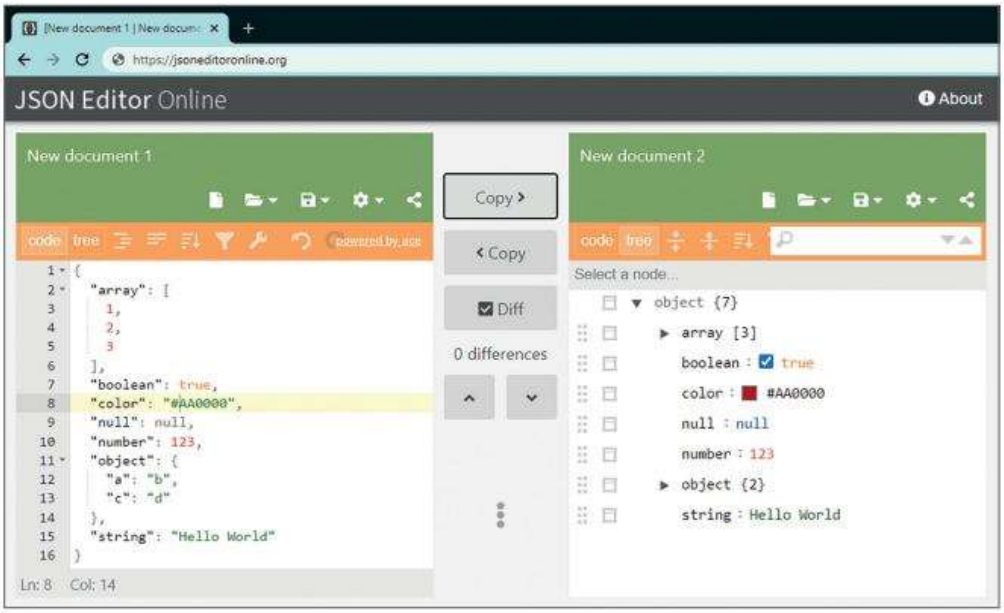


Figura 7.7. *Los editores en línea de JSON son muy completos.*

Por supuesto, existen muchos más editores en línea, con diferentes características y capacidades.

7.4.3. Navegadores

Los navegadores web tienen soporte nativo para la representación de documentos JSON, aunque cada navegador lo hace de forma distinta.

Mozilla Firefox, por ejemplo, permite presentar los datos en texto plano (sin procesar) o en formato JSON, representando la estructura jerárquica y utilizando un código de colores.

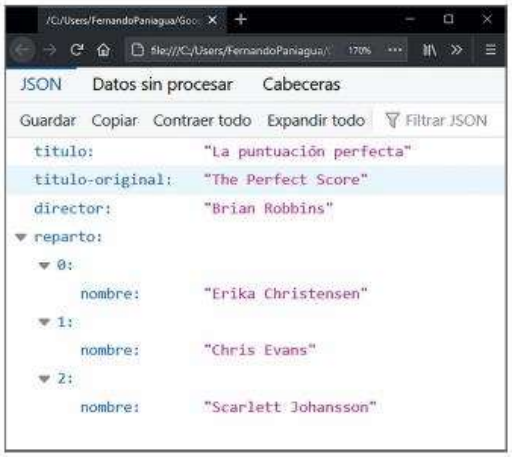
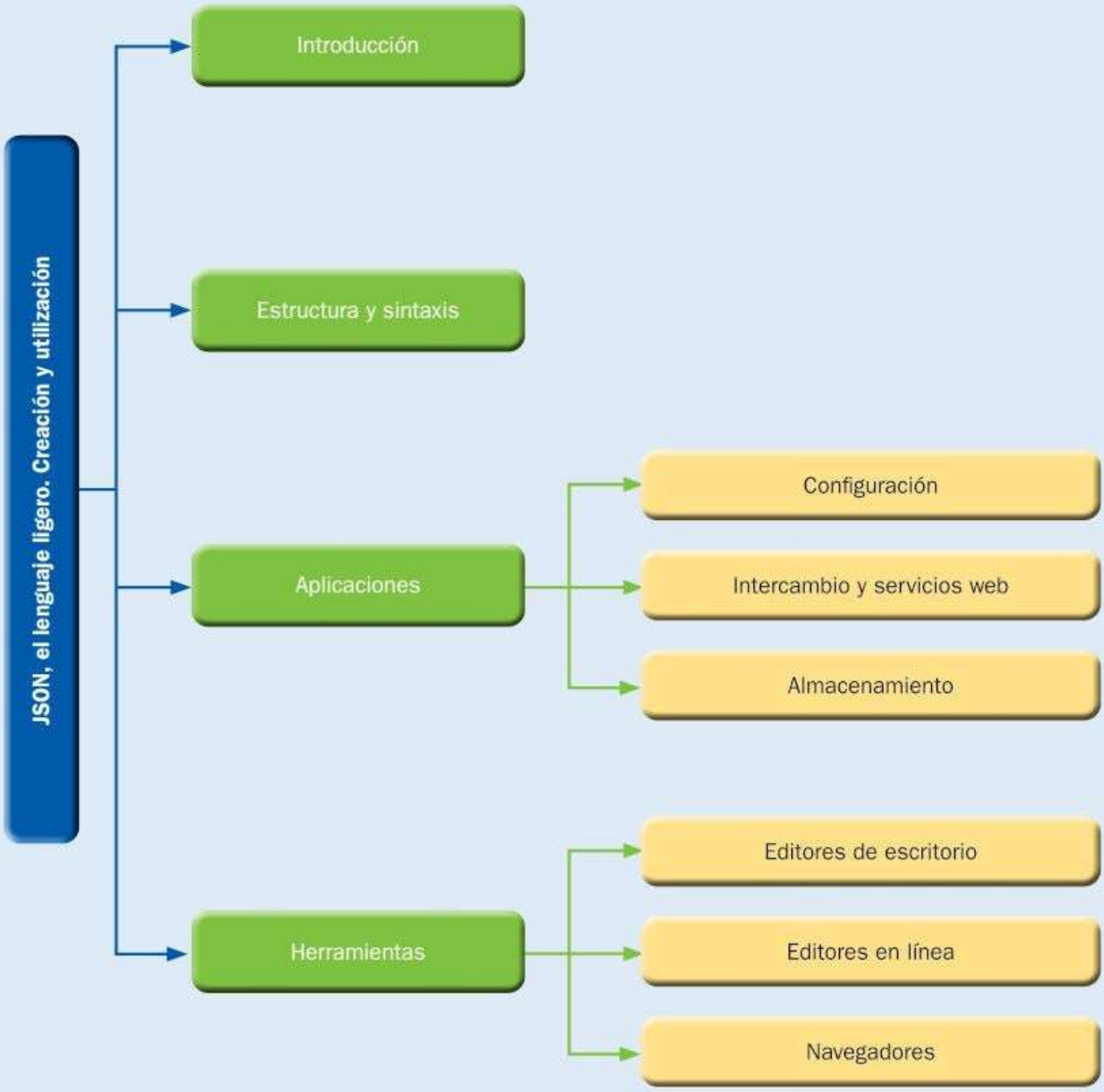


Figura 7.8. *Algunos navegadores saben representar los ficheros JSON.*

© Ediciones Paraninfo



© Ediciones Paraninfo

Actividades de comprobación

7.1. ¿Qué significan las siglas JSON?

a) *Java Simple Object Name.*
b) *JavaScript Object Notation.*
c) *Java Serial Oriented Notation.*
d) *JavaScript Serialized Object Name.*

7.2. ¿De qué lenguaje de programación proviene el formato JSON?

a) Java.
b) JavaScript.
c) Python.
d) C++.

7.3. De manera nativa, ¿qué sistema de validación tiene JSON?

a) DTD.
b) JSON Schema.
c) DTD extendido.
d) JSON no dispone de sistema de validación de forma nativa.

7.4. ¿Cuál de los siguientes es un tipo de elemento de JSON?

a) Clases.
b) Pseudoclases.
c) Objetos.
d) Pseudoobjetos.

7.5. ¿Cuál de las siguientes características es propia de JSON?

a) Los pares clave-valor dentro de un objeto se separan por comas.
b) Los objetos están delimitados por llaves.
c) La separación entre las claves y los valores se realiza con dos puntos.
d) Todas las respuestas son correctas.

7.6. ¿Cuál de las siguientes opciones no se corresponde con un tipo de datos de JSON?

a) Números enteros.
b) Valores lógicos.
c) Fechas.
d) Números decimales.

7.7. ¿Qué símbolo delimita una lista o array en JSON?

a) Comillas.
b) Llave.
c) Coma.
d) Corchete.

7.8. ¿Qué símbolo se utiliza para separar los elementos de una lista o array en JSON?

a) Comillas.
b) Llave.
c) Coma.
d) Corchete.

© Ediciones Paraninfo

7.9. ¿En qué se puede utilizar el formato JSON?

a) Para almacenar la configuración de una aplicación.
b) Para almacenar los datos que intercambiar entre aplicaciones.
c) Para almacenar los datos en bases de datos no relacionales.
d) Todas las respuestas son correctas.

7.10. ¿Con qué herramienta se puede editar un documento JSON?

a) Con un editor de textos sencillo, tipo Bloc de Notas.
b) Con un editor de textos avanzado, tipo Visual Studio Code.
c) Con un procesador de textos, tipo Microsoft Word.
d) Todas las respuestas son correctas.

Actividades de aplicación

7.11. Explica en qué se diferencian JSON y XML.

7.12. Escribe en un documento los pasos que se han de seguir para crear un documento JSON.

7.13. Diseña un documento JSON que almacene la ficha de un libro.

7.14. Diseña un documento JSON que almacene tu *curriculum vitae.*

7.15. Diseña un documento JSON que almacene los libros de una biblioteca organizados por categorías.

7.16. Busca en la web un documento XML y diseña el documento equivalente en JSON. Compara los tamaños en bytes.

7.17. Busca en la web un documento JSON y ábrelo con los siguientes navegadores: Google Chrome, Microsoft Edge, Opera y Firefox. Compara cómo representa el documento cada navegador. ¿Cuál te parece que proporciona la representación más completa?

7.18. Busca e instala una extensión en Google Chrome para que represente los ficheros JSON de forma parecida a como lo hace Firefox. Comprueba su funcionamiento con un fichero JSON alojado en la web.

7.19. Diseña el fichero JSON que deberían intercambiar dos sistemas para transmitir entre ellos correos electrónicos.

7.20. Diseña el fichero JSON que deberían intercambiar dos sistemas para permitir que un taller mecánico comunique un presupuesto a un cliente.

7.21. Escribe en un editor en línea un documento JSON que represente una película. Comprueba que si introduces un error sintáctico, el editor lo identifica correctamente.

7.22. Busca tres editores en línea de ficheros JSON. Pruébalos con el mismo documento JSON. Elige el que te parezca mejor y justifica la decisión.

© Ediciones Paraninfo

Actividades de ampliación

7.23. Busca información referente a las razones por las que se debe utilizar JSON en lugar de XML.

7.24. Halla información sobre las razones por las que se debe utilizar XML en lugar de JSON.

7.25. Busca en internet servicios web que proporcionen los datos en formato JSON.

7.26. Averigua en internet el nombre de conversores en línea que transformen ficheros XML en JSON.

7.27. Estudia en profundidad la sintaxis de JSON y averigua cómo se puede representar un atributo de un elemento XML en este lenguaje.

7.28. Existen dialectos de JSON; BSON y GeoJSON son dos ejemplos. Busca información sobre ellos y averigua para qué se utilizan.

Enlaces web de interés

- **JSON - sintaxis** - https://www.json.org/json-es.html
  (Portal web en español sobre JSON)
- **ECMA - JSON** - http://www.ecma-international.org/publications/files/ECMA-ST/ECMA-404.pdf
  (Estándar ECMA-404 referente a JSON)
- **JSON Editor Online** - https://jsoneditoronline.org
  (Editor online de ficheros JSON)
- **Online JSON Viewer** - http://jsonviewer.stack.hu/
  (Editor online de ficheros JSON)
- **JSON formatter** - https://jsonformatter.org/
  (Editor online de ficheros JSON con corrección automática de formato. Incluye validador, reductor de espacio y conversor)

© Ediciones Paraninfo

UNIDAD 8

Los lenguajes de marcas como herramienta de redifusión de información

Objetivos

- Conocer los fundamentos y la utilidad de los sistemas de redifusión.
- Analizar los formatos más utilizados en redifusión.
- Presentar diversas herramientas para la creación y el consumo de canales de información.
- Enumerar algunos canales de información interesantes de diversos ámbitos.

Contenidos

- 8.1. Introducción a los sistemas de redifusión de información
- 8.2. Formato RSS
- 8.3. Formato Atom
- 8.4. Editores de canales de redifusión
- 8.5. Agregadores de contenidos
- 8.6. Navegadores web y otras herramientas
- 8.7. Canales RSS y Atom

Introducción

La iniciativa en la navegación web parte, tradicionalmente, del usuario. Cuando alguien quiere una información abre el navegador, introduce la URL del sitio web al que quiere acceder y obtiene la información deseada. En este paradigma es el consumidor de la información el único elemento activo de la comunicación.

Los sistemas de redifusión proporcionan un mecanismo que permite que sea el proveedor de los datos el elemento que notifica a los clientes la disponibilidad de nueva información relacionada con sus intereses.

Los sistemas de redifusión más utilizados usan el lenguaje XML como herramienta para la comunicación con los programas empleados por los usuarios para acceder a la información de manera centralizada y automatizada.

8.1. Introducción a los sistemas de redifusión de información

Cuando se accede a la web, normalmente los usuarios abren un navegador, escriben una URL y obtienen la información que desean. Si no conocen la dirección concreta utilizan un buscador, como Google, DuckDuckGo o Yahoo!, para encontrar sitios web candidatos. El buscador ofrece unos resultados y el usuario decide cuál es el más adecuado.

Sea de una forma o de otra, puede que la persona tenga un interés constante en el tiempo en la información publicada en uno o más sitios web. ¿Qué debería hacer para mantenerse informado de las novedades? Tendría que acceder todos los días, o varias veces al día si la frecuencia de actualización es alta, y comprobar de manera personal si hay algún cambio desde la última vez que accedió. Parece evidente que tener información actualizada con este método requiere tiempo y constancia.

Los **sistemas de redifusión web** o **sindicación web** permiten liberar de trabajo a los usuarios de la web al proporcionar un mecanismo de suscripción gracias al cual la información llega a los suscriptores en lugar de tener estos que ir a buscarla. Estos sistemas se basan en canales de redifusión, llamados **fuente web, canal web** o ***web feed.***

Una fuente web es un documento (frecuentemente expresado en lenguaje XML) que contiene información básica sobre elementos de información (noticias o entradas de un blog, por ejemplo), incluyendo las referencias para acceder a los elementos completos.

Al suscribirse a un canal web, los usuarios están permanentemente informados sobre las novedades de este, sin necesidad de tener que ir a buscar la información ni compararla con la obtenida en el último acceso.

Existen dos formatos principales para las fuentes web, ambos basados en XML: **RSS** y **Atom.** La información se publica en uno de estos formatos y su lectura se realiza a través

© Ediciones Paraninfo

de un **agregador** o de un **navegador web,** capaz de realizar las mismas funciones que un agregador mediante el uso de extensiones.

Los ficheros se alojan en un servidor web y el agregador accede a través del protocolo HTTP, obteniendo la información sobre las publicaciones existentes. Los agregadores pueden tener sincronizadas varias fuentes de información (de ahí su nombre), permitiendo al usuario acceder a todos los contenidos desde un único punto. A un grupo de elementos informativos relacionados entre sí y gestionados por un agregador se le conoce coloquialmente como ***feed.***

8.2. Formato RSS

RSS son las siglas de *Really Simple Syndication* («sindicación realmente simple»). Es un **dialecto de XML** utilizado para la distribución de contenidos en la web. Todos los ficheros RSS deben ser conformes a la especificación XML 1.0 del W3C. Es el **formato de fuente web** más popular.

Recuerda

El W3C (World Wide Web Consortium) es un consorcio internacional sin ánimo de lucro que regula los estándares y las recomendaciones relacionadas con la web.

RSS apareció en 1997. **La versión actual es la 2.0** y fue publicada en 2003.

La extensión más habitual utilizada en los ficheros es **.xml**, aunque en ocasiones se emplea **.rss.**

En un archivo RSS se almacenan solo algunos datos del elemento de información, principalmente el título, una descripción y un enlace donde encontrar el elemento de información completo.

La sintaxis es la correspondiente a un documento XML, siendo la estructura para la versión 2.0 la siguiente:

- Un elemento raíz **<rss>** en el que el atributo **version** toma como valor «**2.0**».
- Un elemento **<channel>** por cada canal de suscripción con los siguientes elementos obligatorios y opcionales (todos de tipo texto):

Tabla 8.1. Elementos obligatorios de un canal (<channel>)

Elemento	Descripción
<title>	Contiene el título del canal.
<link>	Contiene el enlace al sitio web correspondiente del canal.
<description>	Contiene una descripción del canal.

© Ediciones Paraninfo

Tabla 8.2. Elementos opcionales de un canal (<channel>)

Elemento	Descripción
<language>	Idioma en el que está escrito el canal.
<copyright>	Aviso relativo a los derechos de autor.
<category>	Una o más categorías a las que pertenece el canal.
<generator>	Nombre del programa con el que se ha creado el canal.

- Una relación de entradas correspondientes a cada elemento informativo (una noticia, un post en un blog, un producto en una tienda de ropa…) marcado con la etiqueta **<item>**, conteniendo los elementos que se muestran en la Tabla 8.3, donde todos son opcionales, aunque debe estar presente al menos <title> o <description>.

Tabla 8.3. Elementos opcionales de un elemento de información (<item>)

Elemento	Descripción
<title>	Contiene el título del elemento informativo.
<link>	Contiene el enlace al elemento informativo completo.
<description>	Contiene una descripción del elemento informativo.
<pubDate>	Contiene la fecha de la publicación del elemento informativo.
<comments>	Si existe, es la URL de la página que contiene los comentarios del elemento informativo.
<author>	Dirección de correo electrónico del autor del elemento informativo.

Recuerda

Especificación completa:

La relación de elementos de RSS expuesta no es completa, en concreto en lo referente a los elementos opcionales. Si quieres conocer el resto de los elementos, lo puedes hacer en los siguientes enlaces:

Elementos opcionales del elemento <channel>:

https://validator.w3.org/feed/docs/rss2.html#optionalChannelElements

Elementos opcionales del elemento <item>:

https://validator.w3.org/feed/docs/rss2.html#hrelementsOfLtitemgt

En el siguiente código XML de ejemplo, se muestra un posible fichero RSS correspondiente a las secciones de Ediciones Paraninfo. Contiene información sobre el canal y dos elementos informativos (<item>):

© Ediciones Paraninfo

```xml
<?xml version="1.0" encoding="UTF-8" ?>
<rss version="2.0">
<channel>
  <title>Ediciones Paraninfo</title>
  <link>https://www.paraninfo.es/</link>
  <description>Web de la Ediciones Paraninfo</description>
  <item>
    <title>Libros de informática</title>
    <link>https://www.paraninfo.es/libros/informatica-N1-0073</link>
    <description>Libros técnicos sobre informática</description>
    <pubDate>Tue, 01 Jan 2021 10:00:00 +0100</pubDate>
  </item>
  <item>
    <title>Libros de electricidad y electrónica</title>
    <link>https://www.paraninfo.es/libros/electricidad-y-electronica-N1-0048</
link>
    <description>Libros técnicos sobre electricidad y electrónica</
description>
    <pubDate>Tue, 03 Jan 2021 10:00:00 +0100</pubDate>
  </item>
</channel>
</rss>
```

Actividad resuelta 8.1

Creación de canal con RSS

Crea un documento RSS con las siguientes características:

- Título: Lenguajes de programación.
- Sitio web: https://www.paraninfo.es/rss.
- Descripción del canal: Canal sobre programación.
- Entradas:
  - Entrada 1:
    - ✓ Título: Python.
    - ✓ Enlace: https://www.paraninfo.es/rss/python.html.
    - ✓ Descripción: el lenguaje Python.
    - ✓Autor: Fernando Paniagua.
  - Entrada 2:
    - ✓ Título: Java.
    - ✓ Enlace: https://www.paraninfo.es/rss/java.html.
    - ✓ Descripción: el lenguaje Java.
    - ✓ Autor: Fernando Paniagua.
  - Entrada 3:
    - ✓ Título: PHP.
    - ✓ Enlace: https://www.paraninfo.es/rss/php.html.

© Ediciones Paraninfo

✓ Descripción: el lenguaje PHP.

✓ Autor: Fernando Paniagua.

Solución

```
<?xml version="1.0" encoding="UTF-8"?>
<rss version="2.0">
  <channel>
    <title>Lenguajes de Programación</title>
    <link>https://www.paraninfo.es/rss</link>
    <description>Canal sobre Programación</description>
    <item>
      <title>Python</title>
      <link>https://www.paraninfo.es/rss/python.html</link>
      <description>El lenguaje Python</description>
      <author>Fernando Paniagua</author>
    </item>
    <item>
      <title>Java</title>
      <link>https://www.paraninfo.es/rss/java.html</link>
      <description>El lenguaje Java</description>
      <author>Fernando Paniagua</author>
    </item>
    <item>
      <title>PHP</title>
      <link>https://www.paraninfo.es/rss/php.html</link>
      <description>El lenguaje PHP</description>
      <author>Fernando Paniagua</author>
    </item>
  </channel>
</rss>
```

Actividad propuesta 8.1

Creación de canal con RSS

Crea un documento RSS para el siguiente canal, correspondiente a una pizzería italiana:

- Título: Pizzas La Serrana.
- Sitio web: www.pizzerialaserrana.it.

Incluye al menos tres elementos referentes a tres tipos de pizza ideados por ti.

8.3. Formato Atom

Atom es el otro gran formato de sindicación de contenidos. Al igual que RSS, está basado en XML y, por lo tanto, un *feed* Atom debe ser un XML bien formado.

La extensiones de los ficheros Atom son **.xml** y **.atom.**

La sintaxis es la correspondiente a un documento XML, siendo la estructura de los ficheros Atom la siguiente:

© Ediciones Paraninfo

- Un elemento raíz **<feed>** en el que se indica que el espacio de nombres es **xmlns= "http://www.w3.org/2005/Atom".** Contiene los elementos indicados en la Tabla 8.4 y, opcionalmente, los elementos indicados en la Tabla 8.5.

Tabla 8.4. Elementos obligatorios y recomendados del elemento <feed>

Elemento	Descripción
<id>	Identifica el canal usando un identificador universal (URI).
<title>	El título del canal.
<updated>	La fecha de la última modificación.
<author>	Puede contener más de un autor. Este elemento debe, a su vez, contener el elemento <name> y puede contener los elementos opcionales <email> y <uri>.
<link>	Define, mediante el atributo **href,** la página web relacionada con el canal.

Tabla 8.5. Elementos opcionales del elemento <feed>

Elemento	Descripción
<category>	La categoría del canal. Puede tener múltiples elementos.
<contributor>	Contiene el nombre de un colaborador, especificado con el elemento <name>. Puede tener múltiples elementos.
<generator>	Identifica el programa utilizado para generar el canal.
<rights>	Contiene información referente a los derechos de propiedad del canal.
<subtitle>	Contiene la descripción del canal.

- Dentro del elemento <feed>, un número variable de elementos **<entry>,** tantos como elementos informativos. Los elementos <entry> están compuestos por los elementos que se muestra en la Tabla 8.6 y, opcionalmente, por los que se muestran en la Tabla 8.7.

Tabla 8.6. Elementos obligatorios y recomendados del elemento <entry>

Elementos	Descripción
<id>	Identifica al elemento usando un identificador universal (URI).
<title>	Título del elemento.
<updated>	Fecha de la última modificación.
<autor>	Nombre del autor. Se indica con el elemento contenido <name>. Puede tener múltiples autores.
<content>	Contiene el enlace o el contenido completo del elemento.
<link>	Referencia a una página relacionada utilizando el atributo **href.**
<summary>	Contiene un breve resumen del elemento.

© Ediciones Paraninfo

Tabla 8.7. Elementos opcionales del elemento <item>

Elementos	Descripción
<category>	Contiene la categoría del elemento utilizando el atributo **term.** Un elemento puede tener múltiples categorías.
<contributor>	Indica un colaborador del elemento mediante el elemento contenido <name>. Puede tener múltiples colaboradores.
<published>	Fecha de la primera creación.
<rights>	Contiene información referente a los derechos de propiedad del elemento.

Recuerda

Especificación completa:

La relación de elementos de Atom expuesta no es completa, en concreto, en lo referente a los elementos opcionales. Si quieres conocer el resto de los elementos lo puedes hacer en los siguientes enlaces:

Elementos opcionales del elemento <feed>:

https://validator.w3.org/feed/docs/atom.html#optionalFeedElements

Elementos opcionales del elemento <entry>:

https://validator.w3.org/feed/docs/atom.html#optionalEntryElements

En el siguiente código XML de ejemplo se muestra un posible fichero Atom correspondiente a las secciones de Ediciones Paraninfo. Contiene información sobre el canal (<feed>) y dos elementos informativos (<entry>):

```
<?xml version="1.0" encoding="utf-8"?>
<feed xmlns="http://www.w3.org/2005/Atom">
  <title>Ediciones Paraninfo</title>
  <link href="https://www.paraninfo.es/"/>
  <updated>2021-01-03T10:00:00Z</updated>
  <author>
    <name>Jack Douglas</name>
  </author>
  <id>http://www.paraninfo.es/feed/</id>
  <entry>
    <title>Libros de informática</title>
    <link href="https://www.paraninfo.es/libros/informatica-N1-0073"/>
    <id>http://www.paraninfo.es/feed/informatica/</id>
    <updated>2021-01-01T10:00:00Z</updated>
    <summary>Libros técnicos sobre informática</summary>
  </entry>
  <entry>
    <title>Libros de electricidad y electrónica</title>
```


© Ediciones Paraninfo

```
    <link href="https://www.paraninfo.es/libros/electricidad-y-
electronica-N1-0048"/>
    <id>http://www.paraninfo.es/feed/electricidadyelectronica/</id>
    <updated>2021-01-03T10:00:00Z</updated>
    <summary>Libros técnicos sobre electricidad y electrónica</summary>
  </entry>
</feed>
```

Actividad resuelta 8.2

Creación de canal con Atom

Crea un documento RSS con las siguientes características:

- Título: Películas sobre tiburones.
- Sitio web: https://www.pelistibu.it/
- Fecha de creación del canal: 3 de septiembre de 2022.
- Autor: Fernando Paniagua.
- Identificador del *feed*: https://www.pelistibu.it/feed/
  - Entrada 1:
    - ✓ Título: Super Tiburón.
    - ✓ Enlace: https://www.pelistibu.it/feed/p1.html.
    - ✓ Identificador: https://www.pelistibu.it/feed/p1.html
    - ✓ Autor: Fernando Paniagua.
    - ✓ Fecha de actualización: 3 de septiembre de 2022.
    - ✓ Resumen: Un enorme tiburón ataca la costa.
  - Entrada 2:
    - ✓ Título: Sharknado.
    - ✓ Enlace: https://www.pelistibu.it/feed/p2.html.
    - ✓ Identificador: https://www.pelistibu.it/feed/p2.html
    - ✓ Autor: Fernando Paniagua.
    - ✓ Fecha de actualización: 4 de septiembre de 2022.
    - ✓ Resumen: Un tornado marino lleva tiburones a la ciudad.

Solución

```
<?xml version="1.0" encoding="utf-8"?>
<feed xmlns="http://www.w3.org/2005/Atom">
  <title>Películas sobre tiburones</title>
  <link href="https://www.pelistibu.it/"/>
  <updated>2022-09-03</updated>
  <author>
    <name>Fernando Paniagua</name>
  </author>
  <id>https://www.pelistibu.it/feed/</id>
  <entry>
    <title>Super Tiburón</title>
```

© Ediciones Paraninfo

```
    <link href="https://www.pelistibu.it/feed/p1.html"/>
    <id>https://www.pelistibu.it/feed/p1.html</id>
    <updated>2022-09-03</updated>
    <summary>Un enorme tiburón ataca la costa.</summary>
  </entry>
  <entry>
    <title>Sharknado</title>
    <link href="https://www.pelistibu.it/feed/p2.html"/>
    <id>https://www.pelistibu.it/feed/p2.html</id>
    <updated>2022-09-04</updated>
    <summary>Un tornado marino lleva tiburones a la ciudad.</summary>
  </entry>
</feed>
```

Actividad propuesta 8.2

Creación de canal con Atom

Crea un documento en formato Atom para el siguiente canal, correspondiente a una empresa familiar dedicada a cultivar y vender productos ecológicos:

- Título: Huertos Ecológicos El Valle.
- Sitio web: www.huertoselvalleecologicos.fr.

Incluye al menos tres elementos referentes a tres productos de huerto ecológicos del catálogo de productos de la empresa.

8.4. Editores de canales de redifusión

Tanto RSS como Atom se construyen sobre documentos XML, por lo que se pueden utilizar los mismos editores válidos para este metalenguaje. Para crear un canal, basta con construir los documentos cumpliendo con la sintaxis y publicarlos en un servidor web.

No obstante, existen aplicaciones para facilitar la creación de los documentos XML correspondientes a los *feeds*. Algunos son sistemas en línea ejecutables a través de un navegador web, como FetchRSS o Feedity. Otros son programas de escritorio, instalables en el ordenador, como RSS Builder.

8.5. Agregadores de contenidos

Un agregador es un programa que permite a los usuarios gestionar las suscripciones a los canales de redifusión. Además de permitir realizar o anular las suscripciones, avisan de la aparición de nuevos elementos informativos, pudiendo configurar cada cuánto tiempo el programa debe revisar el *feed* en busca de novedades.

© Ediciones Paraninfo

Existen muchos agregadores, ya que al basarse en XML no es difícil diseñarlos y programarlos.

Los agregadores pueden ser **de escritorio** y **en línea.** La funcionalidad en ambos casos es parecida, ya que las posibilidades que proporcionan estas dos tecnologías son similares.

Los **agregadores de escritorio** se instalan en el ordenador o en el dispositivo móvil. Algunos de los más usados son NewsFlow, NextGen Reader o RSSOwl.

Figura 8.1. *Los agregadores muestran los elementos de forma visualmente atractiva.*

Los **agregadores en línea** se ejecutan en el navegador y no requieren instalación. Suele ser necesario crear una cuenta para guardar la configuración de los canales de redistribución a los que se está suscrito.

El agregador en línea más utilizado es Feedly. Se puede acceder al servicio creando una cuenta específica o mediante la cuenta de otros servicios web como Google, Twitter o Facebook, por ejemplo. Permite crear *feeds* con mucha facilidad para agregar fuentes de elementos informativos.

Otros agregadores en línea que gozan de popularidad son The Old Reader, Inoreader o Enkel.

© Ediciones Paraninfo

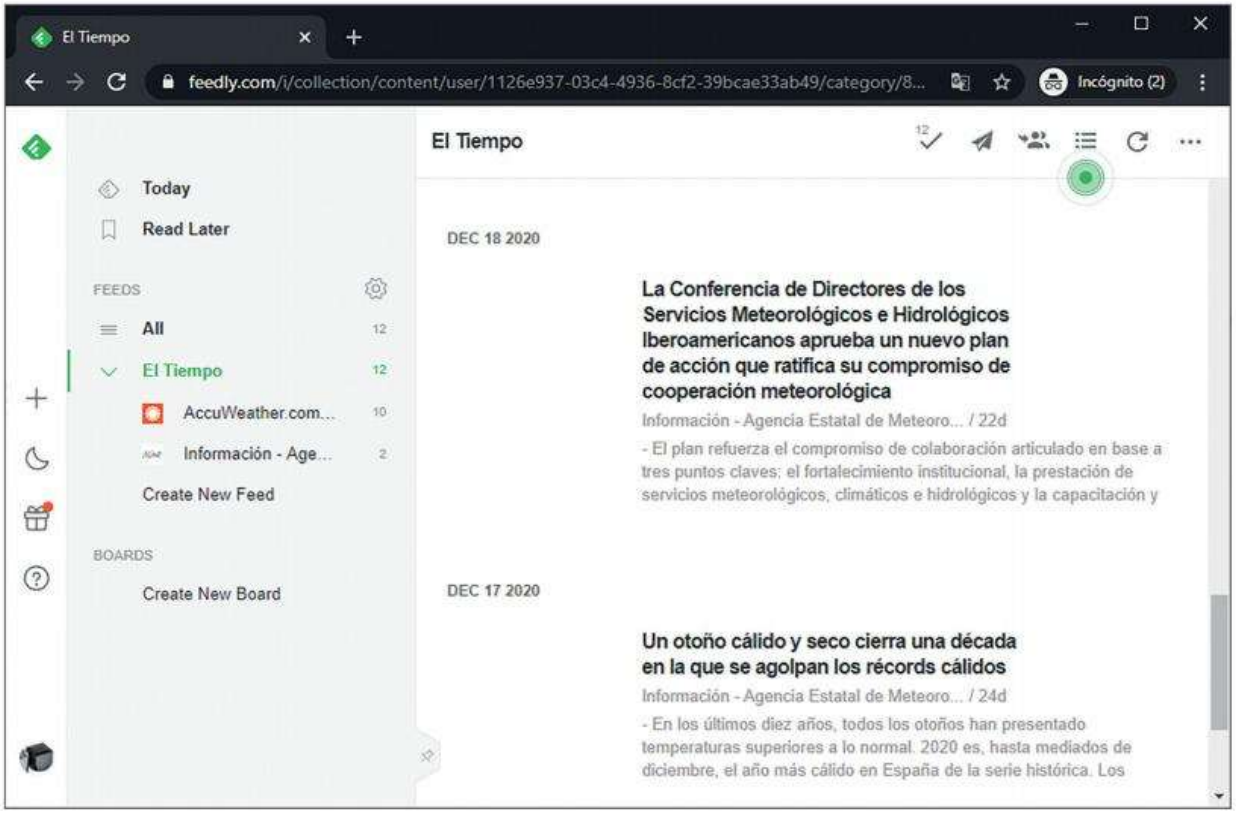


Figura 8.2. *Feedly es uno de los agregadores en línea más populares.*

8.6. Navegadores web y otras herramientas

Además de herramientas específicas para crear *feeds,* existen algunas herramientas que originalmente fueron creadas para otros usos que permiten gestionar las suscripciones a canales de distribución. Los navegadores son un buen ejemplo.

Por defecto, los navegadores web no suelen tener integrado un cliente para suscribirse a los canales de redifusión. Al acceder a un documento RSS o Atom, el navegador muestra el código XML sin realizar ningún proceso.

Afortunadamente, es posible ampliar el funcionamiento de los navegadores con extensiones, convirtiendo así el navegador en un gestor de suscripciones a canales de redistribución.

Google Chrome, por ejemplo, proporciona a través de la tienda **Chrome Web Store** extensiones para gestionar las suscripciones (Figura 8.3).

Por ejemplo, teniendo instalada la extensión **RSS Feed Reader** en Google Chrome, al acceder desde el navegador a un documento RSS o Atom se ofrece la posibilidad de realizar la suscripción.

Una vez realizada la suscripción y configurada debidamente la extensión, el navegador se encarga de avisar cuando hay una novedad en el *feed* de manera automática (Figura 8.5).

© Ediciones Paraninfo

Figura 8.3. *Las extensiones proporcionan nuevas funcionalidades a los navegadores web.*

Figura 8.4. *La suscripción a un feed suele ser tan sencilla como pulsar un botón.*

Figura 8.5. *Algunos agregadores y extensiones muestran las novedades mediante notificaciones.*

Desde el propio navegador se puede acceder a las entradas del canal de redistribución en cualquier momento.

La posibilidad de ampliar mediante extensiones la funcionalidad de un navegador es común a todos los navegadores y suelen existir varias extensiones alternativas, cada una con sus propias características.

© Ediciones Paraninfo

Figura 8.6. *El listado de entradas del canal se puede mostrar de diferentes formas.*

Otro conjunto de herramientas cuya misión principal es distinta, pero permiten integrar lectores de contenidos, son los programas de correo electrónico. Por ejemplo, **Microsoft Outlook** y **Mozilla Thunderbird**, por citar a dos de los clientes de correo electrónico más populares, tienen soporte para suscripciones a canales de distribución de información.

8.7. Canales RSS y Atom

La tecnología que permite la publicación de un canal RSS o Atom es universal y gratuita. Cualquiera puede crear su propio canal de contenidos y publicarlo utilizando estos formatos. Solo es necesario disponer de un servidor web en el que alojar los elementos de información y el fichero correspondiente. Debido a esta sencillez, son muchos los portales web, públicos y privados, que utilizan sistemas de difusión basados en esta tecnología. En este apartado se citarán algunos ejemplos de los innumerables existentes.

Para suscribirse a un canal de un portal web hay que encontrar la página en la que se encuentran los canales (normalmente se publican varios). Se suele identificar el acceso mediante el icono de RSS (Figura 8.7), que también se utiliza para Atom.

Figura 8.7. *El icono de RSS es el símbolo usado de manera genérica para cualquier tipo de feed.*

8.7.1. Prensa general

Los principales medios de comunicación publican canales RSS y Atom. Periódicos de ámbito nacional o regional, como *El País, La Vanguardia, El Periódico de Catalunya, El Periódico de Extremadura, La Voz de Galicia, El Mundo, El Norte de Castilla, La Verdad, El Diario Sur, Diario de Sevilla* o *ABC,* por citar varios ejemplos, tienen sus propios canales.

© Ediciones Paraninfo

Por supuesto, agencias o periódicos internacionales como *The Washington Post, Le Monde* o *BBC News* publican sus contenidos en canales para facilitar el acceso de sus lectores.

8.7.2. Prensa deportiva

De igual manera que la prensa general, la prensa deportiva publica sus noticias a través de canales RSS. Los diarios *Sport, As, Marca* o *Super Deporte,* por ejemplo, disponen de canales con información sobre las noticias.

8.7.3. Inversión

Las bolsas de Madrid y de Barcelona publican en formato RSS información referente a notas de prensa, regulación o avisos relativos a los índices del IBEX. Los diarios *Expansión* o *El Economista* también tienen sus propios canales RSS.

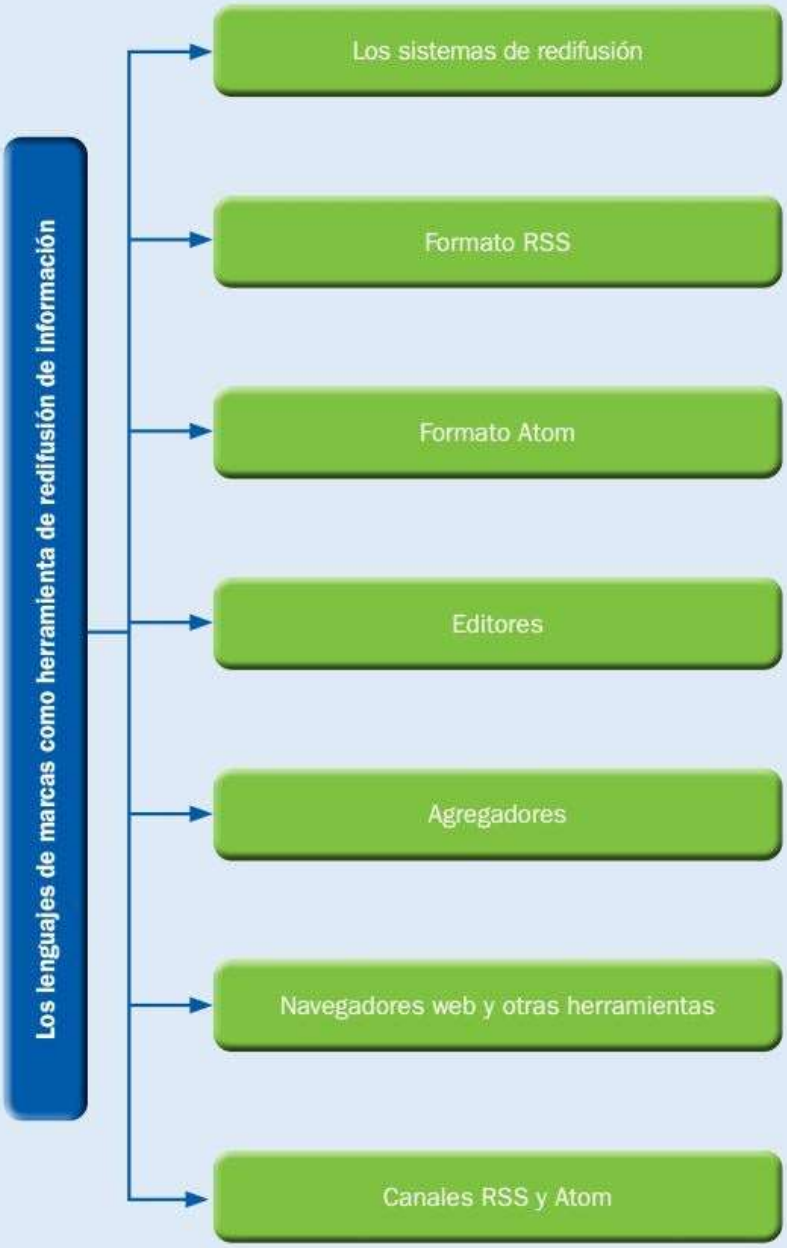
8.7.4. Organismos oficiales y entes públicos

El Ministerio del Interior, el *Boletín Oficial del Estado,* el catálogo de metadatos del Ministerio de Agricultura, Pesca y Alimentación y del Ministerio para la Transición Ecológica y el Reto Demográfico, la Xunta de Galicia, la revista *Tráfico y Seguridad Vial* de la Dirección General de Tráfico, el Banco de España, Radio Televisión Española o Loterías y Apuestas del Estado son algunos de los muchos organismos y entes que tienen sus propios canales RSS y Atom.

8.7.5. AEMET

La Agencia Estatal de Meteorología proporciona varios *feeds* en formato RSS gracias a los cuales se puede obtener información referente a avisos de fenómenos meteorológicos adversos, contratación de personal laboral y convocatoria de oposiciones o a convocatoria de cursos.

© Ediciones Paraninfo



© Ediciones Paraninfo

Actividades de comprobación

8.1. En la navegación web convencional, ¿cuál es el elemento que inicia el flujo de datos?

a) El navegador web.
b) El servidor web.
c) El usuario.
d) El administrador.

8.2. ¿Cómo se llaman los sistemas que proporcionan información a través de canales de datos?

a) Sistemas de canales.
b) Sistemas de redifusión web.
c) Servidor web.
d) Servidor ftp.

8.3. ¿En qué lenguaje de marcas se basan los formatos RSS y Atom?

a) HTML.
b) XHTML.
c) XML.
d) JSON.

8.4. ¿Qué protocolo de comunicación se utiliza en los sistemas de redifusión?

a) FTP.
b) SMTP.
c) HTTP.
d) POP3.

8.5. ¿Qué significan las siglas RSS?

a) *Read System Simple.*
b) *Really Simple Syndication.*
c) *Ready Syndication System.*
d) *Really Simple System.*

8.6. ¿Qué extensión pueden tener los ficheros RSS?

a) .xhtml
b) .xml
c) .html
d) .css

8.7. ¿Cuál es el elemento principal de un documento RSS?

a) <rss>
b) <channel>
c) <title>
d) <description>

8.8. ¿Cuál es el elemento principal de un documento Atom?

a) <entry>
b) <feed>
c) <title>
d) <category>

© Ediciones Paraninfo

8.9. ¿Qué extensión pueden tener los ficheros Atom?

a) .xhtml
b) .xml
c) .html
d) .css

8.10. ¿Qué programas se pueden utilizar para suscribirse a un canal de redifusión?

a) Un cliente de correo electrónico.
b) Un navegador web.
c) Un agregador.
d) Todas las opciones son correctas.

Actividades de aplicación

8.11. Explica qué ventajas ofrece la sindicación de contenidos frente a la navegación tradicional.

8.12. Crea una guía con los pasos necesarios para crear un documento RSS básico para distribuir contenidos.

8.13. Crea una guía con los pasos necesarios para crear un documento Atom básico para distribuir contenidos.

8.14. Busca diferencias entre RSS y Atom y escribe un breve informe enumerándolas.

8.15. Construye un fichero RSS para publicar información relacionada con el ciclo de Formación Profesional que estás cursando. Cada entrada se corresponderá con un módulo formativo.

8.16. Construye un fichero Atom para publicar información relacionada con el ciclo de Formación Profesional que estás cursando. Cada entrada se corresponderá con un módulo formativo.

8.17. Instala una extensión para gestionar tus suscripciones RSS en tu navegador favorito.

8.18. Instala un agregador de contenidos en tu ordenador.

8.19. Busca canales de información RSS o Atom relacionados con alguna de tus actividades favoritas. Suscríbete mediante un agregador.

8.20. Encuentra canales de información RSS relacionados con la NASA y la Agencia Espacial Europea. Descarga alguno de los ficheros de suscripción y analiza su estructura y contenido.

8.21. Halla canales de información RSS relacionados con las principales cadenas de televisión. Suscríbete a alguno de ellos utilizando un agregador.

8.22. Busca dos canales de redifusión que ofrezcan información sobre ofertas de empleo. Comparte los enlaces con tus compañeros y crea un documento que recopile toda la información obtenida.

© Ediciones Paraninfo

Actividades de ampliación

8.23. Obtén información sobre el origen de RSS.

8.24. Busca información referente a las razones por las que se creó Atom.

8.25. Prueba y compara las suscripciones a canales de redistribución realizadas mediante una extensión de un navegador y un agregador en línea.

8.26. Profundiza en la sintaxis de RSS.

8.27. Obtén más información sobre la sintaxis de Atom.

8.28. Estudia en detalle qué información proporciona la Agencia Estatal de Meteorología.

Enlaces web de interés

- **RSS en w3c -** https://validator.w3.org/feed/docs/rss2.html
  (Información sobre RSS2 en w3c)
- **Atom en w3c -** https://validator.w3.org/feed/docs/atom.html
  (Información sobre Atom en w3c)
- **Feedity -** https://feedity.com/
  (Generador de RSS en línea)
- **FetchRSS -** http://fetchrss.com/
  (Generador de RSS en línea)
- **RSS Builder -** https://sourceforge.net/projects/rss-builder/
  (Generador de RSS de escritorio)
- **Newsflow -** https://www.microsoft.com/es-es/p/newsflow/9nblggh58s5r
  (Agregador de escritorio para Windows)
- **RSSOwl -** http://www.rssowl.org/
  (Agregador de escritorio)
- **Feedly -** https://feedly.com/
  (Agregador en línea)

© Ediciones Paraninfo

UNIDAD 9

Sistemas de gestión de información

Objetivos

- Presentar el concepto de sistema de gestión de información.
- Conocer las particularidades de los ERP.
- Aprender en qué puede ayudar un CRM para mejorar la relación con los clientes de una empresa.
- Analizar los sistemas de BI y conocer sus procesos y componentes.
- Descubrir en qué consisten y en qué se diferencian el Cloud Computing y el SaaS.

Contenidos

- 9.1. Introducción a los sistemas de gestión de información
- 9.2. ERP
- 9.3. CRM
- 9.4. BI
- 9.5. Otros sistemas de gestión de información
- 9.6. *SaaS* y *Cloud Computing*

Introducción

Los sistemas de gestión de información son unas herramientas imprescindibles en la actividad de las empresas. Dentro de este término se agrupan varias herramientas que cubren parcial o totalmente los procesos de las empresas.

Las siglas ERP, CRM o BI representan avanzadas tecnologías de vanguardia que dan soporte a los procesos empresariales más importantes, ayudando en actividades cotidianas como la atención al cliente o en otras más excepcionales como la toma de decisiones por parte de la alta dirección.

9.1. Introducción a los sistemas de gestión de información

No existe consenso sobre cuál fue el primer ordenador de propósito general. El *ENIAC* norteamericano, el *Z1* alemán o el *Colossus* británico suelen ser citados como los ganadores de esta carrera, en función del criterio elegido. Sea cual sea el merecedor de este reconocimiento, todos estos ordenadores y sus creadores comparten el mérito de ser pioneros en una tecnología que ha supuesto un impacto enorme en la vida de las personas.

Aunque en su mayoría las primeras computadoras se construyeron en el entorno militar, no pasó mucho tiempo hasta que comenzaron a utilizarse en el entorno empresarial. La rapidez a la hora de realizar los cálculos, la fiabilidad o la reducción de costes (pese a que al comienzo de la era de la informática los ordenadores no eran precisamente económicos) fueron argumentos suficientemente atractivos para hacer que las grandes empresas fijasen su atención en esta incipiente tecnología.

En esta etapa temprana de la informática, aún muy lejana de la era de internet, las operaciones que eran capaces de realizar los ordenadores eran bastante sencillas y no fue hasta la aparición de los lenguajes de programación de alto nivel como Cobol, Pascal, Fortran o C cuando se comienzan a crear de forma cotidiana aplicaciones empresariales relacionadas con la gestión.

El término **sistemas de gestión de información** referido al entorno empresarial se aplica a un conjunto muy extenso de aplicaciones informáticas, tanto que es difícil encontrar una definición precisa. No obstante, si se reduce al mínimo común el alcance se puede afirmar que los sistemas de gestión de información **son programas informáticos diseñados para dar soporte a cualquiera de los procesos que se realizan en una empresa.**

Los procesos a los que hace referencia esta definición son de naturaleza diferente: productivos, comerciales, logísticos o de gestión de recursos humanos son solo algunos ejemplos. La dificultad a la hora de precisar qué es un sistema de gestión de información se relaciona con el alcance del sistema en sí.

Por ejemplo, una macro para calcular un presupuesto en una hoja de cálculo en Microsoft Excel podría considerarse en un sistema de gestión de información. También podría

© Ediciones Paraninfo

considerarse como tal un programa en Python que, cada cierto tiempo, se conecte a una base de datos, compruebe el stock de los productos en el almacén y envíe un correo electrónico cuando haya riesgo de desabastecimiento. Otro ejemplo sería una aplicación que gestione el capital humano de una empresa, analizando los perfiles de los empleados, la calidad de su trabajo actual y las necesidades de la empresa para establecer programas de formación.

Como se puede comprobar, existen candidatos a sistema de gestión de información muy simples y otros mucho más sofisticados. Resulta difícil poner la línea divisoria entre qué es y qué no es un sistema de gestión de información, ya que, como ocurre en casi todos los ámbitos, las fronteras suelen ser difusas.

Se considera **sistema de gestión de información aplicado al entorno empresarial** una aplicación que cubre un proceso empresarial completo, lo que excluiría a los tres ejemplos citados anteriormente.

Todos los sistemas de gestión de información (en realidad, prácticamente todos los sistemas del tipo que sean) pueden ejecutarse en máquinas locales o en otras pertenecientes a terceras empresas, mediante computación en la nube (conocida como *Cloud Computing*).

En esta unidad se van a presentar algunos de los más importantes sistemas de gestión de información, describiendo su funcionalidad, enumerando sus características y presentando algunos ejemplos concretos utilizados en el mundo de las empresas. También se expondrán las diferencias, ventajas e inconvenientes existentes entre las diferentes alternativas de ejecución.

9.2. ERP

Cuando una empresa se digitaliza, las primeras aplicaciones que desarrolla permiten calcular nóminas, hacer balances, obtener listados de deudores o gestionar el stock de un almacén. Las empresas las van creando en función de las necesidades, el presupuesto o la capacidad de asumir riesgos y de innovar. Son aplicaciones independientes, islas programáticas sin conexión entre sí, lo que termina inevitablemente suponiendo una limitación.

Las aplicaciones de una empresa deben establecer canales de comunicación entre sí: las ventas de un producto repercuten en el stock de este en el almacén; el pago de una nómina en la tesorería; una avería en una máquina en la capacidad de reposición; la calidad en la construcción de los productos en el canal de atención al cliente. Al disponer de aplicaciones independientes, a veces creadas en lenguajes distintos e incluso estar ejecutándose en máquinas separadas, el proceso de integración resulta costoso en tiempo, esfuerzo y recursos.

Para salvar este inconveniente y facilitar la integración entre los diferentes sistemas existen los ERP, una solución consistente en concebir al conjunto de las aplicaciones de una empresa como un todo en lugar de tratar cada sistema como una unidad independiente y autónoma.

© Ediciones Paraninfo

Un ERP (*Enterprise Resource Planning,* «sistemas de planificación de recursos empresariales», en castellano) es un sistema informático que integra los múltiples procesos específicos que hay en una empresa.

Estos procesos son, por ejemplo, control de producción, gestión de compras y ventas, gestión de recursos humanos, control de almacén, contabilidad, control financiero o atención al cliente.

Los ERP en el entorno empresarial comienzan a utilizarse a partir de los años sesenta del siglo XX, pero es a partir de la década de los setenta cuando comienzan a coger impulso, siendo empresas pioneras SAP y Oracle.

Figura 9.1. *Los ERP están compuestos de módulos relacionados entre sí.*

En desarrollo de software, se considera **producto** a un sistema creado para cubrir una necesidad, pero sin estar dirigido a una empresa en concreto. Estos productos son posteriormente adaptados al cliente, que implanta personal propio de la empresa o consultores externos, expertos en el producto en cuestión.

Los ERP son considerados **productos** de software. Los consultores que se dedican a la implantación y mantenimiento de este tipo de sistemas tienen perfiles muy especializados, tanto desde el punto de vista técnico, relacionado con el sistema en sí, como desde el punto de vista funcional, relacionado con los diferentes módulos. Por ejemplo, es habitual que un consultor de un módulo financiero, además de conocer dicho módulo en el ERP concreto en el que es experto, también posea conocimientos avanzado sobre finanzas.

Nota técnica

Se conoce como **implantación** al proceso que permite utilizar un sistema software en un entorno concreto. Su alcance es más amplio que el término **instalación,** más relacionado con un proceso de copia de ejecutables y librerías y de realizar pequeñas configuraciones.

© Ediciones Paraninfo

9.2.1. Características de los ERP

Un ERP es un sistema modular, que cubre gracias a dichos módulos todos los procesos de negocio, pero es un único sistema. Un conjunto de aplicaciones dispares que ayudan en la realización de determinados procesos y que se comunican entre sí no es un ERP. Para considerar a un sistema ERP tiene que cumplir ciertas características.

Así, un ERP debe ser:

- **Funcional.** Un ERP debe proveer soluciones para todos los procesos del negocio.
- **Modular.** Los módulos se deben poder integrar de manera opcional e independiente.
- **Centralizado.** La base de datos debe ser única y los datos no han de estar replicados en múltiples fuentes.
- **Robusto.** Debe incorporar mecanismos que garanticen la fiabilidad de los datos.
- **Seguro.** Tiene que incorporar mecanismos que garanticen que el acceso a la información solo se va a permitir a las personas autorizadas.
- **Mantenible.** El sistema debe poder mantenerse, mejorarse y ampliarse. Es importante a la hora de elegir un ERP mantenido por una tercera empresa que esta sea solvente y fiable. De lo contrario, cualquier imprevisto podrá convertirse en un gran problema.
- **Adaptable.** Los ERP son soluciones genéricas, construidas como productos lo suficientemente flexibles como para poder ser integrados en casi cualquier compañía. Esto implica que la implantación de un ERP en una empresa requiere de un proceso de instalación, configuración y adaptación al entorno de esta.

Nota técnica

Se conoce como *suite* a un conjunto de aplicaciones que se comercializan conjuntamente. Suelen ser aplicaciones independientes que cubren distintos procesos de negocio. En ocasiones se confunden los conceptos ERP y suite, pero no son lo mismo, ya que tienen diferentes características.

9.2.2. Tipos de ERP

Como ocurre con todos los sistemas, los ERP se pueden categorizar de muchas maneras. Una de las clasificaciones más útiles guarda relación con el grado de personalización o adaptación a la empresa. Los más genéricos suelen ser más económicos, pero obligan a la empresa a adaptar sus procesos al sistema. Los más personalizados suelen exigir más inversión, pero se adaptan a los requisitos de la empresa.

Según este criterio, los tipos de ERP son:

- **Genéricos.** Son sistemas diseñados como un producto modular, lo que facilita la escalabilidad. No están pensados para ninguna empresa en concreto. Suelen ser sencillos de implantar y la opción más económica. El inconveniente que presentan es que pueden no tener la capacidad de adaptarse a los procesos de la empresa.

© Ediciones Paraninfo

- **Configurables o parametrizables.** El núcleo del sistema está basado en múltiples parámetros modificables, permitiendo adaptar los módulos a los procesos de la empresa. En esta opción el coste de la implantación suele ser alto, ya que la adaptación puede ser compleja y requerir de un gran esfuerzo especializado.
- **A medida.** Consiste en desarrollar el ERP desde cero, basado en los requisitos específicos de la empresa a la que va destinado. Es una opción que requiere una gran inversión y solo debe elegirse cuando las otras opciones no son viables.

9.2.3. Beneficios aportados por los ERP

Muchas empresas realizan sus operaciones sin ERP. Las razones son diversas: el desconocimiento, la falta de capacidad técnica o económica para hacer la transición a este tipo de sistemas, inversiones previas no amortizadas o, simplemente, que debido al tipo de actividad de la empresa el coste no supera a los beneficios.

Los ERP son interesantes para empresas medianas y grandes que realizan muchos procesos de forma interna. Es en este tipo de empresas donde los ERP son útiles y aportan valor.

Los ERP aportan importantes beneficios a las empresas. A continuación, se citan algunos de estos beneficios:

- Optimiza los procesos de gestión.
- Mejora de la productividad.
- Simplificación de las tecnologías de la información de la compañía (TI).
- Mejora en los procesos de decisión como consecuencia de la centralización de la información y del acceso a los datos en tiempo real.
- Reduce los costes de gestión de la información.
- Facilitan la escalabilidad del sistema.
- Permiten incrementar la seguridad de la información.

La decisión de implantar un ERP a la empresa y la elección de este son decisiones complejas, basada en las necesidades de la compañía, la capacidad del departamento de sistemas o el presupuesto disponible.

9.2.4. Fabricantes de ERP más populares

Existe un gran número de ERP tanto comerciales como gratuitos. Se citan a continuación algunas de las empresas más populares junto con referencias a sus productos.

SAP

Es la empresa líder indiscutible de los ERP. Fundada en 1972 en Alemania, goza de gran prestigio. Su producto principal es SAP ERP, compuesto por módulos que cubren la mayoría de los procesos empresariales.

© Ediciones Paraninfo

Oracle

Es el gran competidor de SAP, además del gran líder en sistemas de gestión de bases de datos relacionales, y fue fundada en Estados Unidos en 1977. Tras sucesivas adquisiciones de empresas del sector, Oracle dispone de tres soluciones de ERP: Oracle JD Edwards Enterprise One, Oracle People Soft y Oracle E-Business One. Además, como muchas otras empresas del sector, Oracle está apostando por las soluciones basadas en la computación en la nube y el SaaS (*Software as a Service* o «software como servicio») con los productos ERP Oracle NetSuite y Oracle ERP Cloud.

SAGE

Ubicada en el Reino Unido y fundada en 1981, es otra de las grandes empresas que disponen de productos ERP. También es muy conocida por su software de contabilidad, líder en este sector. Dispone de las soluciones ERP Sage X3 y Sage 200, apostando firmemente por prestar los servicios en la nube.

Microsoft

El líder indiscutible en sistemas operativos para ordenadores personales también tiene su propia solución ERP. Microsoft fue fundada en 1975 en Estados Unidos y desarrolla su negocio en múltiples áreas de la tecnología. Ha creado el célebre sistema operativo Windows, la familia de consolas Xbox, la suite ofimática Office o herramientas para el desarrollo de software. Microsoft Dynamics 365 es su producto ERP.

IFS

IFS es una empresa sueca creada en 1983. Su ERP se denomina IFS Applications y se considera como uno de los más innovadores del mercado.

9.2.5. ERP gratuitos

Además de los grandes productos comerciales, existen alternativas gratuitas y que suelen ser suficientes para empresas de tamaño pequeño y mediano.

Esta relación de ERP incluye algunos de dichos productos.

ERPNext

Este ERP tiene gran tradición. Posee módulos para controlar almacenes, aspectos financieros, ventas, etc. La versión gratuita debe instalarse en los ordenadores de la compañía y no dispone de soporte, que se puede contratar aparte. También ofrecen una versión en la nube con coste.

© Ediciones Paraninfo

Odoo

Odoo destaca por la sencillez de su administración, ya que tiene una interfaz amigable y simple. Dispone de una versión en la web que requiere el pago de una licencia o descargar el código fuente para implantarlo en las propias máquinas de la empresa.

Apache OFBiz

Este ERP es muy personalizable, permitiendo adaptarse a empresas medianas y grandes que dispongan de recursos y personal cualificado para ello. Es gratuito y tiene la suficiente trayectoria para convertirlo en un producto maduro. Está escrito en Java y su código es modificable.

WebERP

Como su propio nombre indica, está basado en la web y utiliza tecnologías relacionadas, como HTML5, JavaScript y PHP. Se utiliza a través de un navegador y es relativamente sencillo de modificar y mantener.

9.3. CRM

La relación de los clientes con las empresas puede ser compleja. Un cliente es un comprador satisfecho de un producto, pero también el insatisfecho que efectúa una devolución o reclamación. Hay clientes puntuales y recurrentes. Un cliente se pone en contacto con las empresas a través de muchos canales: personalmente, por teléfono, a través de correo electrónico e incluso por WhatsApp. Puede solicitar un presupuesto por medio de un formulario web y llamar después por teléfono para aclarar algún aspecto. A un cliente le pueden contactar varios empleados de departamentos distintos y por diferentes canales para hacer el seguimiento de una incidencia.

Figura 9.2. *Un CRM gestiona de manera unificada los canales de relación con los clientes.*

© Ediciones Paraninfo

Esta relación mal gestionada puede provocar en los clientes frustración e insatisfacción y podría desencadenar en su pérdida. Por ejemplo, en el transcurso de una petición de información, si cada departamento involucrado pide todos los datos al cliente una y otra vez, provocará hartazgo y, posiblemente, rechazo. Lo mismo ocurrirá si un cliente hace un pedido, lo modifica más tarde, pero esta modificación no se recoge. También si un empleado se compromete a informarle y, tras terminar su turno, no lo hace la persona que lo reemplaza en su puesto de trabajo.

Un CRM es un tipo de sistema de gestión de información específico para atender toda esta casuística. Las siglas CRM significan *Customer Relationship Management* o, lo que es lo mismo, «gestión de relación con clientes». Es una herramienta fundamental en la gestión del activo más importante para cualquier empresa: sus clientes.

9.3.1. Características de los CRM

Algunas de las características que debe cumplir un CRM son las siguientes:

- **Gestión de contactos.** Tener el control absoluto de qué interacción ha sucedido entre la empresa y el cliente es la principal función de estos sistemas. La gestión de los contactos aglutina toda la información del cliente, que debe ser fácil de acceder.
- **Clasificación de clientes.** Las acciones de marketing son eficaces si el destinatario es el correcto. Los sistemas CRM permiten clasificar (o calificar) a los clientes para ayudar en la segmentación.
- **Unificación de canales.** Todos los canales de comunicación deben ser uno para el cliente. Un CRM debe unificar el flujo de interacciones para que haya una continuidad en la comunicación.
- **Automatización y seguimiento de los flujos de trabajo.** Existen acciones relacionadas con los clientes que forman parte de un proceso. Por ejemplo, una acción comercial consistente en efectuar llamadas a los clientes para ofrecerles un producto y realizar el posterior seguimiento requiere de organización y control. Un CRM debe ser capaz de automatizar este proceso para que se desarrolle de forma completa y correcta.
- **Gestión de las redes sociales.** No todos los CRM incluyen esta funcionalidad, pero debido a la importancia adquirida por las redes sociales, se está convirtiendo en una característica que tener en cuenta.

9.3.2. Beneficios aportados por los CRM

Los CRM aportan los siguientes beneficios a las empresas:

- **Facilitan la toma de decisiones.** Al unificar los datos de los clientes en un único punto, resulta más sencillo obtener información sobre su comportamiento, necesidades, nivel de satisfacción o capacidad económica. Gracias a esta información, las decisiones podrán ser más eficaces.
- **Incrementan las oportunidades de venta.** Al conocer más sobre los clientes (gustos, historial de compras, perfil…), es más sencillo acertar al ofrecerles nuevos

© Ediciones Paraninfo

productos. La acción de venta se puede integrar en los canales de comunicación, permitiendo ser mucho más selectivos a la hora de gestionar nuevas oportunidades de venta.

- **Posibilitan realizar acciones de marketing eficaces**, gracias a la clasificación de los clientes.
- **Ayudan en el proceso de mejora continua.** Los datos relativos a la satisfacción de los clientes, las incidencias recogidas o las cuestiones recibidas permiten detectar los puntos débiles de la empresa y facilitar su mejora.

9.3.3. CRM más populares

Algunos de los sistemas CRM más populares son los siguientes:

Salesforce

Salesforce es una plataforma basada en la nube que proporciona a todos los departamentos de la empresa que tienen relación con los clientes una visión unificada de estos. Es uno de los líderes del sector.

Microsoft Dynamics 365 CRM

Forma parte del ecosistema *Dynamics 365* de Microsoft. Admite ejecución en la nube y es altamente escalable. Incluye módulos para atender a los departamentos de marketing, ventas y atención al cliente, entre otros.

Zoho CRM

Es uno de los CRM más populares. Tiene una extensa red de clientes en todo el mundo, lo que lo convierte en una alternativa fiable. La automatización de flujos de trabajo, la generación de informes y el uso de la inteligencia artificial para ayudar en los procesos de ventas son algunos de sus puntos fuertes. Dispone de una versión gratuita con algunas restricciones pero válida para pequeñas empresas.

SAP Customer Experience

El fabricante alemán de sistemas dispone de una solución para la gestión de la relación con los clientes. Contiene módulos como *SAP Customer Data Cloud* o *SAP Marketing Cloud,* que ayudan en los procesos de conversión de clientes y en las acciones de marketing.

Oracle Customer Experience (CX)

El gigante de las bases de datos también dispone de su propio CRM. Su mayor activo es la experiencia de la empresa en el manejo de grandes cantidades de datos, lo que asegura un rendimiento óptimo en cualquier circunstancia.

© Ediciones Paraninfo

9.4. BI

Las siglas BI significan *Business Intelligence* y se puede traducir como «inteligencia de negocio» o «inteligencia empresarial», aunque lo habitual es que las referencias a este conjunto de tecnologías se hagan a través de su nombre en inglés.

BI está formado por distintas tecnologías que tienen un objetivo común: ayudar en el proceso de dirección de la empresa mediante la unificación y tratamiento de los datos generados como consecuencia de su actividad.

Por ejemplo, una empresa de transporte de mercancías gasta una cantidad de dinero anual en combustible. Este gasto viene determinado por el medio de transporte utilizado, su nivel de carga, el cálculo de la ruta o incluso el estilo de conducción del transportista. Gracias al BI se pueden cotejar todos los datos relacionados con la actividad, analizarlos y modificar los procesos para reducir los costes.

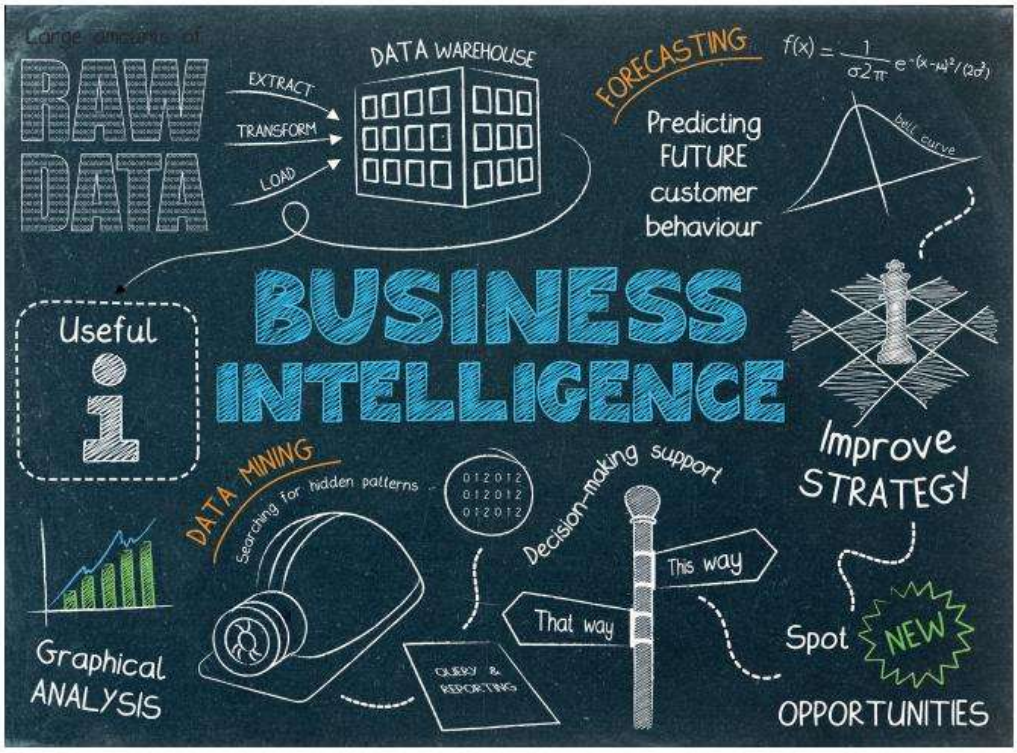


Figura 9.3. *El uso de BI facilita la toma de decisiones empresariales.*

Los procesos y subsistemas en los que se divide el BI son los siguientes:

- ETL. Proceso de extracción de los datos en bruto *(raw data)* de los sistemas transaccionales, transformación y almacenamiento en los sistemas *Data Warehouse.*
- *Data Warehouse.* Almacén de los datos procesados obtenidos por el proceso ETL.
- Utilización de los datos del *Data Warehouse* para la toma de decisiones:
  - OLAP y *Data Mining.* Permiten organizar los datos en estructuras multidimensionales para generar información imperceptible mediante técnicas estadísticas y de inteligencia artificial.

© Ediciones Paraninfo

– Cuadro de mandos. Resumen visual del estado de la empresa.

– Informes. Presentación gráfica y numérica del estado de la empresa.

– Otros sistemas de representación.

A continuación, se explican brevemente cada uno de estos procesos y subsistemas.

9.4.1. ETL

Los datos generados en las operaciones diarias de una compañía pueden ser de unas dimensiones enormes, de cientos de miles o de millones de sucesos susceptibles de ser recogidos en las bases de datos donde se almacenan las operaciones.

Con este volumen de datos, cualquier consulta en tiempo real que implique muchos registros puede requerir un tiempo de espera que en ocasiones no es aceptable.

Por ejemplo, una empresa que tenga cientos de sedes, miles de clientes y decenas de miles de ventas diarias genera un número importante de registros. ¿Cuánto tiempo tardará un miembro de la dirección de la empresa en obtener los datos referentes a las ventas de una de las sedes durante el mes en curso? ¿Y de los clientes de entre 30 y 32 años en todas las sedes durante un mes determinado? ¿Será rápida la generación de un informe con las ventas de todas las sedes durante las primeras dos horas de la mañana acumuladas por semanas? Las respuestas a todas estas preguntas serán similares y tendrán que ver con el tiempo que se tiene que esperar a que se ejecute la consulta por parte del usuario: con muchos registros el sistema no proporcionará una respuesta rápida.

En el mundo empresarial, especialmente en la alta dirección, los tiempos de respuesta son vitales para que la toma de decisiones sea eficaz, y hay procesos que por sus dimensiones no pueden proporcionar resultados de forma rápida en tiempo real. Para solucionar este problema existe el proceso conocido como ETL.

ETL significa *Extract, Transform and Load* («extraer, transformar y cargar») y es el proceso que carga de datos los *Data Warehouse*.

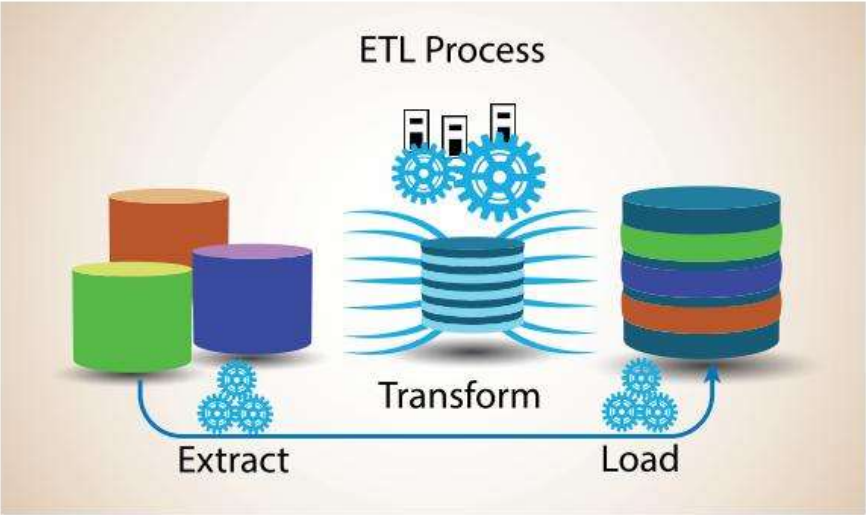


Figura 9.4. *El proceso ETL está compuesto por tres subprocesos: extracción, transformación y carga.*

© Ediciones Paraninfo

El proceso de ETL toma como entrada los datos de la operación cotidiana de la empresa (extracción), los convierte en datos agrupados (transformación) y los almacena (carga) en el sistema de almacenamiento que se utiliza para la toma de decisiones.

De esta forma, las respuestas a las consultas de ejemplo que se citaban anteriormente se habrían realizado de forma previa a su utilización, con lo que los datos ya estarían disponibles en el momento en el que se desee acceder a ellos. Lógicamente los datos no están actualizados en tiempo real, ya que el proceso ETL se realiza de manera diaria, semanal o con la frecuencia que sea adecuada, pero las decisiones de dirección no se toman con, por ejemplo, las ventas del día de hoy, sino con el histórico de ventas de la última semana, mes o año.

9.4.2. *Data Warehouse*

Es un componente fundamental en BI. Traducido como «almacén de datos», es el conjunto de los datos de una compañía que permiten la correcta toma de decisiones.

Los almacenes de datos no tienen datos en tiempo real, sino datos de hechos que ya han ocurrido y que son, por lo tanto, permanentes.

Figura 9.5. *Los almacenes de datos o Data Warehouse se alimentan de los datos de las operaciones de la empresa.*

En un almacén de datos, por ejemplo, se almacena información sobre las ventas que se han realizado en la compañía, agrupadas y procesadas para facilitar el acceso orientado a la toma de decisiones.

El proceso de obtención de estos datos es complejo y requiere de soluciones y herramientas específicas.

9.4.3. OLAP

OLAP son las siglas de *Online Analytical Processing* («procesamiento analítico en línea»). Es la tecnología que, dentro del *Business Intelligence,* facilita el acceso a los datos almacenados en los *Data Warehouse* para alimentar el proceso de *Data Mining.*

© Ediciones Paraninfo

Figura 9.6. *La tecnología OLAP permite representar los datos desde distintos puntos de vista.*

OLAP estructura la información en lo que se conoce como **cubo OLAP**, una estructura multidimensional perfecta para representar hechos sucedidos en diferentes ubicaciones, segmentos temporales o relacionados con distintas entidades del sistema. Para este tipo de procesos, son más ágiles que las estructuras de datos relacionales convencionales.

9.4.4. *Data Mining*

La minería de datos, conocida como *Data Mining*, es el proceso que consiste en buscar patrones repetitivos en grandes volúmenes de datos, normalmente obtenidos de los almacenes de datos o *Data Warehouses.*

Figura 9.7. *Utilizando Data Mining se puede obtener información no evidente.*

© Ediciones Paraninfo

El objetivo de esta búsqueda es descubrir tendencias y reglas que expliquen la evolución de los datos para poder anticipar comportamientos y tomar medidas basadas en la predicción.

La minería de datos utiliza técnicas estadísticas y de inteligencia artificial para encontrar los patrones. Es importante tener claro que **el *Data Mining* proporciona información que no se conoce *a priori*,** y no es el resultado de una búsqueda de información dirigida como ocurre con otras técnicas.

En ocasiones se confunde este término con las técnicas de extracción de datos (ETL).

9.4.5. Cuadro de mandos

El término *Dashboard* o cuadro de mandos no es específico del *Business Intelligence,* pero sí está íntimamente relacionado.

Los procesos incluidos en las técnicas de *Business Intelligence* reducen considerablemente el volumen de los datos que hay que consultar para conocer el estado de la empresa y tomar las mejores decisiones. No obstante, en función del tamaño y de las operaciones de la empresa, el número de datos que hay que consultar puede ser importante. Además, son datos numéricos que requieren de un análisis para convertirlos en información y este proceso, cuando lo realiza una persona, puede requerir bastante esfuerzo y tiempo.

Un cuadro de mandos es, en *Business Intelligence,* una síntesis gráfica del estado de la empresa.

Figura 9.8. *Un cuadro de mandos permite conocer de un simple vistazo el estado de una empresa o un proceso.*

© Ediciones Paraninfo

El cuadro de mandos permite consultar, de forma inmediata, evoluciones, alertas, tendencias y estados de casi cualquier elemento de una compañía.

Los sistemas de *Business Intelligence* proporcionan cuadros de mandos altamente configurables para facilitar el acceso a la información de la empresa por parte de la alta dirección.

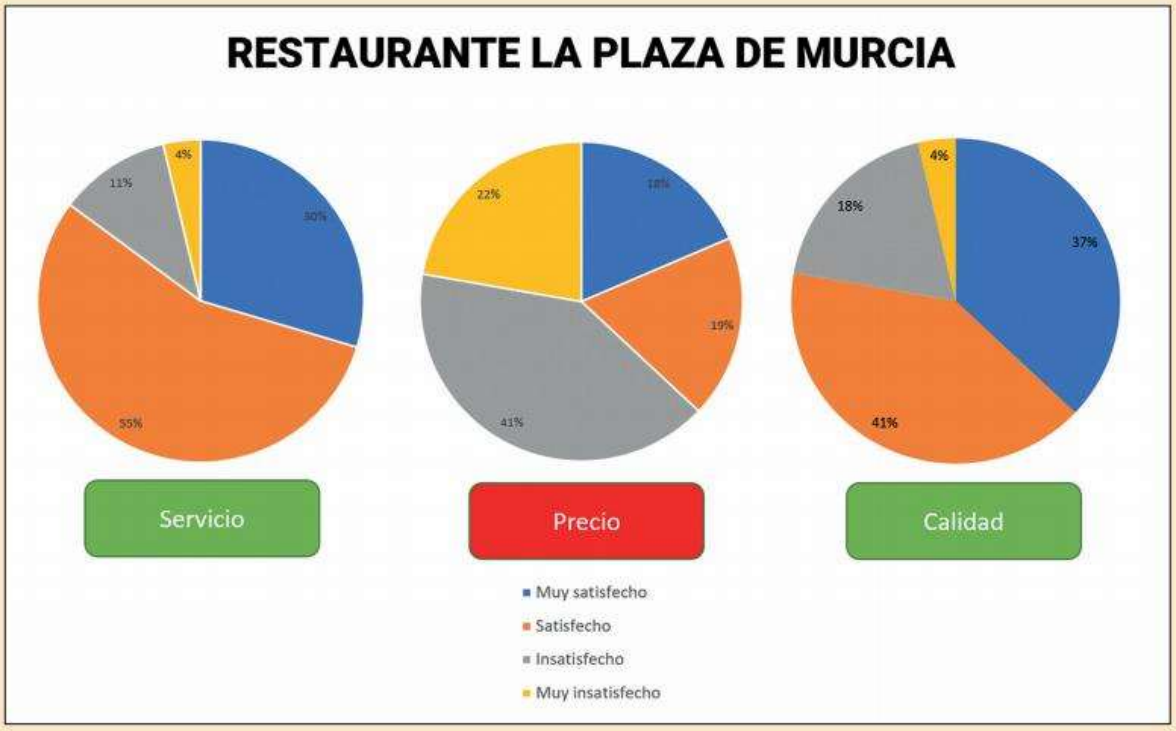
Actividad resuelta 9.1

Diseño de un cuadro de mandos

Dada una empresa que dispone de varios restaurantes, se quiere disponer de un cuadro de mandos que, para cada uno de ellos, muestre el nivel de satisfacción de los clientes en tres aspectos: servicio, precio y calidad. Mediante un rectángulo de color verde se mostrarán los nombres de los aspectos en los que el porcentaje de clientes muy satisfechos o satisfechos sea superior a los insatisfechos o muy insatisfechos. En aquellos aspectos en los que los porcentajes de valoración negativos sean mayores que los positivos, se mostrará el rectángulo de color rojo. De esta manera, con un simple vistazo se podrá saber en qué aspectos hay que mejorar cada uno de los restaurantes.

En esta actividad se debe diseñar el cuadro de mandos de uno de estos restaurantes.

Solución



Actividad propuesta 9.1

Diseño de un cuadro de manos

Una tienda de tecnología vende cinco familias de productos: teléfonos móviles, patinetes eléctricos, videoconsolas, ordenadores y drones.

© Ediciones Paraninfo

La facturación mensual de cada una de estas familias en el mes de febrero es:

- Teléfonos móviles: 23.000 €.
- Patines eléctricos: 3.500 €.
- Videoconsolas: 31.000 €.
- Ordenadores: 28.000 €.
- Drones: 10.130 €.

Se debe diseñar un cuadro de mandos en el que poder consultar, mediante un sistema de colores, el volumen de facturación de cada familia de productos en un mes determinado para conocer qué productos aportan más o menos a la facturación.

Las reglas de presentación son las siguientes:

- Si la familia de productos supone menos del 10% de la facturación mensual, el color es rojo.
- Si la familia de productos supone entre el 10% y el 20% de la facturación mensual, el color es naranja.
- Si la familia de productos supone más del 20% de la facturación mensual, el color es verde.

Recuerda que el objetivo del cuadro de mandos es saber, de forma inmediata, qué está bien y qué no lo está en la empresa.

Se puede utilizar cualquier herramienta para hacer el diseño, incluso un papel y bolígrafos de colores.

9.4.6. Informes

La generación de informes (actividad también conocida como *Reporting*) es otra de las alternativas de visualización de datos del *Business Intelligence*.

Suele presentar la información con tablas, diagramas y gráficas. Se utilizan herramientas que son sencillas de configurar, permitiendo que los usuarios finales puedan generar sus propios informes sin requerir ayuda de personal especializado.

9.4.7. Sistemas de BI más populares

Algunos de los sistemas de BI existentes en el mercado y que gozan de popularidad se enumeran a continuación.

Microsoft Power BI

Permite que los usuarios finales generen sus propios informes y cuadros de mandos gracias a una sencilla interfaz y a una usabilidad muy optimizada.

Soluciones Oracle

Oracle Business Intelligence es el sistema de BI de Oracle. Surge como resultado de la adquisición por parte de esta empresa de otras compañías con productos de BI.

© Ediciones Paraninfo

Oracle Analytics es otra de las alternativas de *Business Intelligence* de Oracle. Integra mecanismos de aprendizaje automático (ML, *Machine Learning*) y de inteligencia artificial (AI, *Artificial Intelligence*).

MicroStrategy

Esta empresa norteamericana, fundada en 1989, es una de las referencias en BI. Tiene unas herramientas muy potentes para el análisis de datos y la creación de informes.

TIBCO Spotfire

TIBCO Sportfire es una plataforma de análisis de datos empresariales que pertenece a la empresa TIBCO. Goza de gran prestigio y entre sus características más relevantes se incluye la búsqueda de información utilizando lenguaje natural.

Otras soluciones BI

Hay más soluciones populares en el mercado, respaldadas por empresas de gran prestigio. Algunas de ellas son IBM Cognos, SAP AG Business Objects, Qlik Sense, SAS y Tableau.

9.5. Otros sistemas de gestión de información

En la actividad de la gestión de información en el ámbito empresarial existen más sistemas. En el siguiente listado se citan los más importantes:

- SCM. *Supply Chain Management* o «administración de la cadena de suministros» ayuda en la gestión de la cadena de suministros para optimizar los procesos de aprovisionamiento, almacenamiento y distribución de la mercancía. Los sistemas de gestión de la logística pueden formar parte de estos sistemas.
- SARH y SIRH. Sistemas de administración de recursos humanos y sistemas de información de recursos humanos. Proporcionan apoyo a la gestión de los recursos humanos.

9.6. *SaaS* y *Cloud Computing*

Los términos **SaaS** y ***Cloud Computing*** no son específicos de los sistemas de gestión de información, pero tienen tanta relevancia que es importante aclarar qué son, en qué se diferencian, qué ventajas aportan y qué inconvenientes pueden presentar.

El *Cloud Computing* o «computación en la nube» es un término cuyo alcance es muy genérico. Consiste en utilizar los recursos computacionales proporcionados por otra empresa: CPU, almacenamiento, esquemas de seguridad, copias, etc. Es, en cierto modo, la contraposición a la computación en las máquinas propias de la compañía. Normalmente el

© Ediciones Paraninfo

sistema de facturación se basa en el uso que se hace de la plataforma, siendo mayor a medida que aumenta el consumo de recursos.

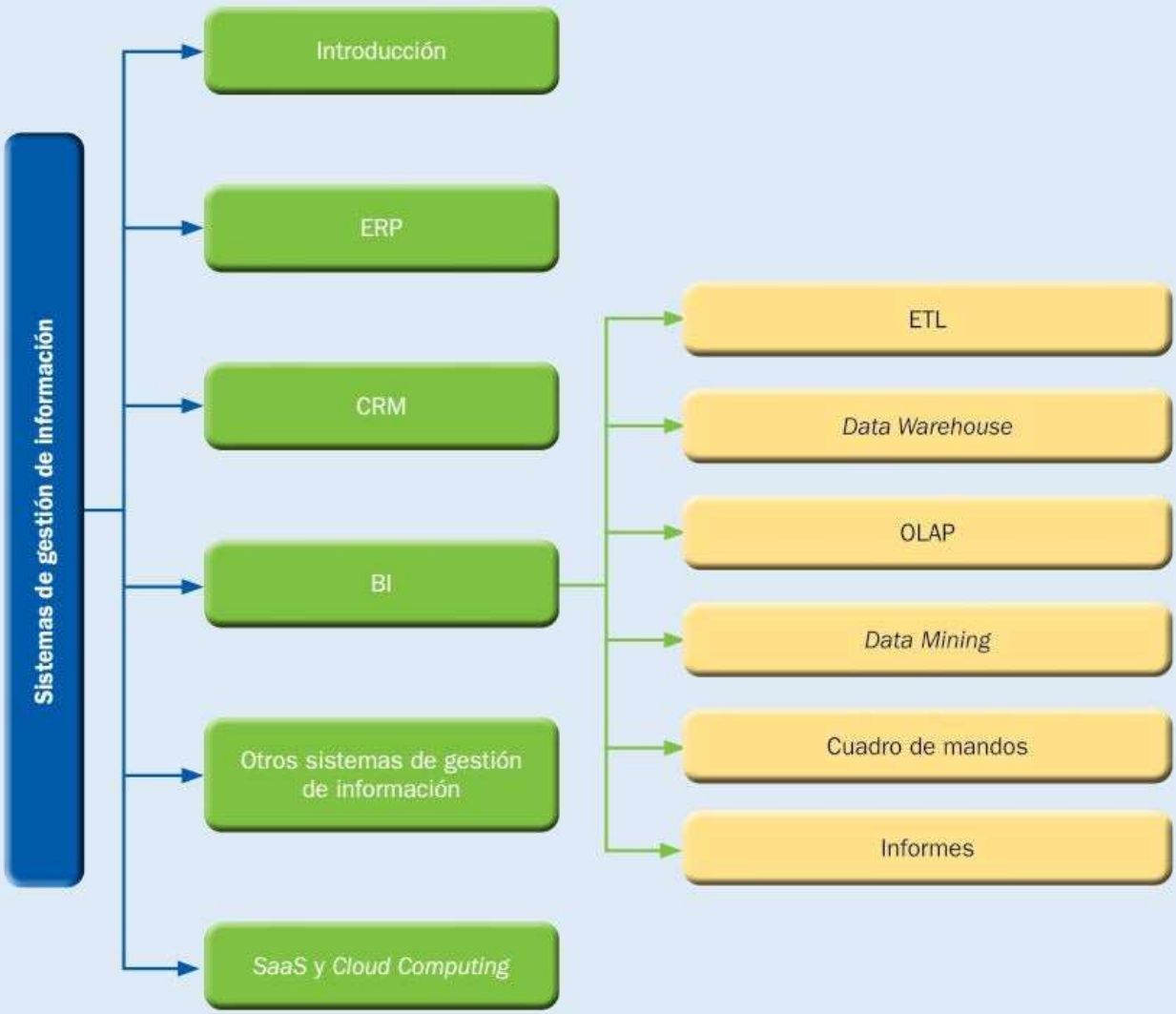
La computación en la nube presenta varias ventajas. Una de ellas es la reducción de costes. Al pagar por el uso, los gastos siempre serán en proporción a la actividad. También se reducen los costes al no tener que disponer de un **centro de proceso de datos** (CPD) propio ni de personal especializado en administración de sistemas. La seguridad es otra de las grandes ventajas, ya que las empresas proveedoras de los servicios *Cloud* suelen contar con personal especializado y mucha experiencia. La disponibilidad del servicio es otro de los apartados en los que la nube presenta mayores ventajas, ya que las compañías que prestan estos servicios suelen garantizar un gran porcentaje de disponibilidad que, en el caso de los sistemas propios, resulta difícil conseguir.

El *SaaS* (*Software As A Service* o «software como servicio») es un modelo de computación en la nube en la que la empresa proveedora proporciona el servicio. Por ejemplo, una empresa puede proporcionar un sistema de control de almacén a otra empresa, proporcionándole tanto la infraestructura física como la lógica, incluido el sistema software y el mantenimiento de este. El cobro del servicio suele hacerse en función del consumo, lo que suele implicar una reducción de costes. **Los fabricantes de sistemas de gestión de información empresarial suelen ofrecer sus productos en la modalidad de *SaaS*.**

Figura 9.9. *El SaaS permite dimensionar los sistemas y los costes en función de la actividad.*

En ambos casos, las desventajas principales tienen dos perspectivas: la legal y la relativa a la seguridad. Existen regulaciones en España y en Europa que tienen que ver con la gestión de datos de carácter personal que pueden impedir que estos sean almacenados en sistemas ajenos a la empresa. Es muy importante comprobar que no se incumple ninguna normativa al contratar servicios en la nube que estén alojados en terceros países. En relación con la seguridad, muchas empresas almacenan en sus sistemas de gestión información crítica para su negocio y no ven con buenos ojos que dicha información esté fuera de su estricto control. Además, se puede correr el riesgo de que la empresa que proporciona el servicio quiebre, modifique su política de precios repentinamente o decida suspender el servicio. Hay que valorar también la fiabilidad de la empresa que proporciona los servicios en la nube, ya que, en cierto modo, se está delegando una parte muy importante del negocio de una compañía.

© Ediciones Paraninfo



© Ediciones Paraninfo

Actividades de comprobación

9.1. Indica qué afirmación referente a los sistemas de gestión de información es correcta:

a) Dan soporte a los procesos empresariales.
b) Cubren procesos empresariales completos.
c) Se pueden ejecutar en máquinas propias de la empresa o en la nube.
d) Todas las opciones anteriores son correctas.

9.2. ¿Qué significan las siglas ERP?

a) *Enterprise Resource Project.*
b) *Enterprise Resource Planning.*
c) *Enterprise Remote Planning.*
d) *Enterprise Remote Project.*

9.3. ¿Cuál de las siguientes empresas es pionera en los sistemas ERP?

a) SAP.
b) Microsoft.
c) Apple.
d) Hewlett Packard.

9.4. ¿Cuál de las siguientes características no es propia de un ERP?

a) Modular.
b) Mantenible.
c) Utiliza múltiples repositorios de datos.
d) Adaptable.

9.5. ¿Cuál es el tipo de ERP que normalmente resulta más costoso?

a) Genérico.
b) Configurable.
c) A medida.
d) Todos son iguales de costosos.

9.6. ¿Qué significan las siglas CRM?

a) *Client Relationship Management.*
b) *Customer Relationship Management.*
c) *Customer Relationship Model.*
d) *Client Relative Management.*

9.7. ¿En qué ayuda un CRM a una empresa?

a) Permite conocer el estado de las facturas pendientes de pago.
b) Unifica los canales de comunicación con los clientes.
c) Informa de las unidades de un producto disponibles en el almacén.
d) Todas las respuestas son correctas.

9.8. ¿Cuáles de los siguientes términos se corresponde con BI?

a) ETL.
b) *Data Warehouse.*
c) Cuadro de mandos.
d) Todas las respuestas son correctas.

© Ediciones Paraninfo

9.9. ¿Qué significan las siglas ETL?

a) Extracción, transferencia y carga.
b) Estudio, transformación y carga.
c) Extracción, transformación y carga.
d) Estudio, transformación y compromiso.

9.10. ¿Qué es un *Data Warehouse*?

a) Es una herramienta de extracción de información.
b) Es un almacén de datos.
c) Es un sistema de generación de informes.
d) Es un sistema de análisis de riesgos.

9.11. Las estructuras de datos de los cubos OLAP:

a) Son de una única dimensión.
b) Son de dos dimensiones.
c) Son multidimensionales.
d) Son árboles.

9.12. La minería de datos:

a) Responde a preguntas en lenguaje natural.
b) Busca información concreta en una base de datos.
c) Encuentra patrones para extraer información desconocida.
d) Es lo mismo que ETL.

9.13. Un cuadro de mandos:

a) Permite manejar una fábrica.
b) Permite manejar todos los sistemas de una empresa.
c) Proporciona una vista gráfica y resumida del estado de una empresa y de sus procesos.
d) Proporciona una herramienta de control remoto de los almacenes.

9.14. Los sistemas *SaaS:*

a) Son sistemas en la nube.
b) Los proporcionan terceras empresas.
c) Facilitan la escalabilidad.
d) Todas las respuestas son correctas.

Actividades de aplicación

9.15. Explica con tus propias palabras qué es un sistema de gestión de información.

9.16. Enumera los tipos de ERP y busca ejemplos de cada uno de dichos tipos. Comparte los ejemplos con tus compañeros para formar una lista más completa.

9.17. Solicita información por correo electrónico a alguno de los fabricantes de ERP. Indica que eres un estudiante y que quieres saber dónde puedes aprender a utilizar su sistema. Haz un breve informe con la información obtenida.

© Ediciones Paraninfo

9.18. Explica con tus propias palabras qué es un CRM.

9.19. Busca información concreta sobre, al menos, dos CRM y comprueba que cumplen con las características propias de estos sistemas.

9.20. Explica con tus propias palabras la relación que existe entre el proceso ETL y el *Data Warehouse*.

9.21. Suponiendo que se dispone de una base de datos con información referente a las ventas de productos cosméticos, explica con tus propias palabras en qué consistiría el proceso ETL que obtuviese las ventas mensuales. Indica el origen de los datos, el filtro que aplicar, la operación de transformación y el destino de los datos.

9.22. Dibuja en un papel un cuadro de mandos que muestre información sobre el inventario de un almacén, teniendo en cuenta que almacena tres productos diferentes y que uno de ellos tiene menos unidades de las convenientes.

9.23. Explica con tus propias palabras la diferencia entre ETL y *Data Mining*.

9.24. Explica con tus propias palabras la diferencia entre *Data Warehouse* y OLAP.

9.25. Diseña en papel un informe en el que se vea la siguiente información relativa a un producto: ventas semanales por ciudad y porcentaje de impagos sobre el total de ventas. Utiliza tablas y gráficos.

9.26. Si fueses el director de tecnología de un hospital, ¿qué impedimento encontrarías a la hora de utilizar *Cloud Computing*? Justifica la respuesta.

Actividades de ampliación

9.27. Profundiza en el concepto de sistema de gestión de información.

9.28. Busca información referente a la formación en el ERP de SAP. Ten en cuenta que tiene diferentes módulos, por lo que tendrás que elegir sobre qué módulo realizar la formación.

9.29. Busca en YouTube un tutorial sobre algún CRM de los citados en esta unidad.

9.30. Obtén en la web más información sobre los procesos ETL.

9.31. Encuentra ejemplos en la web de cuadros de mandos y analiza sus elementos.

9.32. Investiga sobre los tres principales proveedores de *Cloud Computing* y averigua cómo puedes formarte en sus tecnologías.

© Ediciones Paraninfo

Enlaces web de interés

- **SAP** - https://www.sap.com/spain/index.html
  (Portal web de SAP)
- **Oracle** - https://www.oracle.com/es/index.html
  (Portal web de Oracle)
- **Sage** - https://www.sage.com/es-es/
  (Portal web de Sage)
- **Microsoft** - https://www.microsoft.com/es-es
  (Portal web de Microsoft)
- **IFS** - https://www.ifs.com/es/
  (Portal web del ERP de IFS)
- **ERPNext** - https://erpnext.com/
  (Portal web del ERP gratuito y de código abierto ERPNext)
- **ODOO** - https://www.odoo.com/es_ES/
  (Portal web de Odoo)
- **OFBIZ** - https://ofbiz.apache.org/
  (Portal web de Apache OFBIZ)
- **WEBERP** - http://www.weberp.org/
  (Portal web del ERP weberp)
- **Salesforce** - https://www.salesforce.com/es/
  (Portal web de Salesforce)
- **ZOHO** - https://www.zoho.com/es-xl/crm/
  (Portal web del CRM Zoho)
- **Microstrategy** - https://www.microstrategy.com/es
  (Portal web de Microstrategy)
- **TIBCO** - https://www.tibco.com/es/products/tibco-spotfire
  (Portal web de Tibco)

© Ediciones Paraninfo

Lenguajes de marcas y sistemas de gestión de información

Los lenguajes de marcas permiten el funcionamiento de la web y la construcción de los sistemas de información, que son imprescindibles para dar soporte a los procesos empresariales.

En este libro se desarrollan los contenidos del módulo profesional Lenguajes de marcas y sistemas de gestión de información de los Ciclos Formativos de Grado Superior de Desarrollo de Aplicaciones Multiplataforma, Desarrollo de Aplicaciones Web y Administración de Sistemas Informáticos en Red, pertenecientes a la familia profesional de Informática y Comunicaciones.

Lenguajes de Marcas y Sistemas de Gestión de Información se presenta como una herramienta para descubrir y aprender a utilizar los principales lenguajes de marcas, así como una puerta de entrada al amplio mundo de los sistemas de gestión de información en el ámbito empresarial. Se pueden destacar los siguientes contenidos:

- **Lenguajes de marcas para la web:** HTML, XHTML y CSS, como principales representantes.
- **Lenguajes de marcas para el intercambio de información y soporte a los sistemas:** XML, incluyendo transformaciones y varias de las tecnologías relacionadas con este lenguaje, así como sus herramientas de validación y JSON, como alternativa ligera a XML.
- **Herramientas:** para la edición, el almacenamiento y el acceso a la información representada por los distintos lenguajes de marcas.
- **Sistemas de gestión de información:** conocer los principales tipos de sistemas de gestión de información utilizados en las empresas.

El autor, **Fernando Paniagua Martín,** es ingeniero informático y docente en escuelas de formación, universidades y centros de formación profesional, dentro de la familia de Informática y Comunicaciones, desde hace más de dos décadas.

RECURSOS PARA EL PROFESORADO

- Programación didáctica
- Solucionario
- Videotutoriales sobre actividades prácticas
- Presentación de aula
- Examina
- LDP (Libro Digital Proyectable)

Estos recursos son exclusivos para el profesorado que confirme con nuestro Departamento de Promoción la adopción de este título como libro de texto en el aula.

Paraninfo
ciclos formativos
www.paraninfo.es

ISBN: 978-84-283-4059-5
También en Ebook
9 788428 340595