TEMA 1.
CARACTERÍSTICAS DE LOS LENGUAJES DE MARCAS

Lenguajes de Marcas
CFGS DAW 1

Autor: Pascual Ligero

Revisado por:

Fco. Javier Valero – franciscojavier.valero@ceedcv.es

2019/2020

Versión:190920.1642

Licencia

Nomenclatura

A lo largo de este tema se utilizarán distintos símbolos para distinguir elementos importantes dentro del contenido. Estos símbolos son:

 Importante

 Atención

 Interesante

ÍNDICE DE CONTENIDO

UD01. CARACTERÍSTICAS

1. ORDENADOR E INFORMACIÓN

1.1 Formas de representar información en el ordenador

El ordenador es una máquina digital, por lo tanto sólo es capaz de representar información utilizando el sistema binario de numeración. Esto obliga a que, para poder almacenar información en un ordenador, previamente haya que codificarla en forma de números binarios.

El problema de los números binarios es que están muy alejados del ser humano; es decir, que las personas no estamos capacitadas para manejar información en binario. Nosotros usamos sistema decimal para los números y formas de representación mucho más complejas para otra información (como el texto, las imágenes, la música,...)

Sin embargo actualmente un ordenador es capaz de manejar información de todo tipo: música, imágenes, texto,...Esto es posible porque se ha conseguido que casi cualquier tipo de información sea codificable en binario.

Los seres humanos tenemos la capacidad de diferenciar claramente lo que es un texto de una imagen, lo que es un número de una canción,...Pero en un ordenador todo es más complicado , porque todo es binario.

Desde los inicios de la informática la codificación (el paso de información humana a información digital) ha sido problemática debido a la falta de acuerdo en la representación. Pero hoy día ya tenemos numerosos estándares.

Fundamentalmente la información que un ordenador maneja son Números y Texto. Pero curiosamente a nivel formal se consideran datos binarios a cualquier tipo de información representable en el ordenador, que no es texto (imagen, sonido, vídeo,...), aunque como ya hemos comentado, en realidad toda la información que maneja un ordenador es binaria, incluido el texto.

1.2 Datos en forma de texto y datos binarios

Datos binarios

Cualquier dato que no sea texto, se considera dato binario. Por ejemplo: música, vídeo, imagen, un archivo Excel, un programa,...

La forma de codificar ese tipo de datos a su forma binaria es muy variable. Por ejemplo en el caso de las imágenes, cada punto (píxel) de la imagen se codifica utilizando su nivel de rojo, verde y azul. De modo que una sola imagen produce millones de dígitos binarios.

En cualquier caso sea cual sea la información que estamos codificando en binario, para poder acceder a dicha información, el ordenador necesita el software que sepa como decodificar la

misma, es decir saber qué significa cada dígito binario para traducirle a una forma más humana. Eso sólo es posible utilizando el mismo software con el que se codificó o bien otro software pero que sea capaz de entender la información codificada.

Texto

El texto es quizá la forma más humana de representar información. Antes de la llegada del ordenador, la información se transmitía mediante documentos o libros en papel. Esa forma de transmitir es milenaria y sigue siendo la forma más habitual de transmitir información entre humanos; incluso con la tecnología actual aplicaciones como twitter, whatsapp,... siguen usando el texto como formato fundamental para transmitir información.

En cuanto apareció la informática como una ciencia digital, apareció también el problema de cómo codificar texto en forma de dígitos binarios para hacerlo representable en el ordenador. La forma habitual ha sido codificar cada carácter en una serie de números binarios. De modo que por ejemplo el carácter A fuera por ejemplo 01000001 y la B el 01000010.

El problema surgió por la falta de estandarización, la letra A se podía codificar distinto en diferentes ordenadores y así nos encontrábamos con un problema al querer pasar datos de un ordenador a otro.

Poco a poco aparecieron estándares para intentar que todo el hardware y software codificara los caracteres igual.

El código ASCII

El problema de la codificación de texto que hacía incompatibles los documentos de texto entre diferentes sistemas, se palió cuando se ideó en 1967 un código estándar por parte de la ANSI, la agencia de estándares norteamericana, dicho código es el llamado ASCII (American Standard Code for Information Interchange, código estándar americano para el intercambio de información).

El código utiliza el alfabeto inglés (que utiliza caracteres latinos) y para codificar todos los posibles caracteres necesarios para escribir en inglés se ideó un sistema de 7 bits (con 7 bits se pueden representar 128 símbolos, suficientes para todas las letras del alfabeto inglés, en minúsculas y mayúsculas, caracteres de puntuación, símbolo s especiales e incluso símbolos de control).

El código ASCII es el siguiente:

Tabla de Códigos ASCII (0-127)

Carácteres no imprimibles				Carácteres imprimibles								
Nombre	Dec	Hex	Car.	Dec	Hex	Car.	Dec	Hex	Car.	Dec	Hex	Car.
Nulo (null)	0	00	NUL	32	20	Espacio	64	40	@	96	60	`
Inicio de cabecera	1	01	SOH	33	21	!	65	41	A	97	61	a
Inicio de texto	2	02	STX	34	22	"	66	42	B	98	62	b
Fin de texto	3	03	ETX	35	23	#	67	43	C	99	63	c
Fin de transmision	4	04	EOT	36	24	$	68	44	D	100	64	d
Enquiry	5	05	ENQ	37	25	%	69	45	E	101	65	e
Acknowledge	6	06	ACK	38	26	&	70	46	F	102	66	f
Beep (sonido)	7	07	BEL	39	27	'	71	47	G	103	67	g
Backspace	8	08	BS	40	28	(	72	48	H	104	68	h
Tabulador horizontal	9	09	HT	41	29	)	73	49	I	105	69	i
Salto de línea	10	0A	LF	42	2A	*	74	4A	J	106	6A	j
Tabulador vertical	11	0B	VT	43	2B	+	75	4B	K	107	6B	k
Salto de página	12	0C	FF	44	2C	,	76	4C	L	108	6C	l
Retorno de carro	13	0D	CR	45	2D	-	77	4D	M	109	6D	m
Shift fuera	14	0E	SO	46	2E	.	78	4E	N	110	6E	n
Shift dentro	15	0F	SI	47	2F	/	79	4F	O	111	6F	o
Escape línea de datos	16	10	DLE	48	30	0	80	50	P	112	70	p
Control dispositivo 1	17	11	DC1	49	31	1	81	51	Q	113	71	q
Control dispositivo 2	18	12	DC2	50	32	2	82	52	R	114	74	r
Control dispositivo 3	19	13	DC3	51	33	3	83	53	S	115	73	s
Control dispositivo 4	20	14	DC4	52	34	4	84	54	T	116	74	t
Acknowledge negativo	21	15	NAK	53	35	5	85	55	U	117	75	u
Síncronismo	22	16	SYN	54	36	6	86	56	V	118	76	v
Fin bloque transmitido	23	17	ETB	55	37	7	87	57	W	119	77	w
Cancelar	24	18	CAN	56	38	8	88	58	X	120	78	x
Fin medio	25	19	EM	57	39	9	89	59	Y	121	79	y
Sustituto	26	1A	SUB	58	3A	:	90	5A	Z	122	7A	z
Escape	27	1B	ESC	59	3B	;	91	5B	[	123	7B	{
Separador archivos	28	1C	FS	60	3C	<	92	5C	\	124	7C	|
Separador grupos	30	1E	RS	62	3E	>	94	5E	^	126	7D	}
Separador unidades	31	1F	US	63	3F	?	95	5F	_	127	7F	DEL

Dec. Decimal
Hex. Hexadecimal
Car. Caracter

Ejemplo: al carácter 'J' le corresponde el código ASCII decimal 74 que en hexadecimal se representa como 4A, y en binario como 01001010 (este último no se encuentra en la tabla)

Pero, en países con lenguas distintas del inglés, surgió el problema de que parte de los símbolos de sus alfabetos quedaban fuera del ASCII (como la letra eñe).

Por ello se diseñaron códigos de 8 bits que añadían 128 símbolos más y así aparecieron los llamados códigos ASCII extendidos. En ellos, los 128 símbolos primeros son los mismos de la tabla ASCII original y los 128 siguientes se corresponden a símbolos extra. Así por ejemplo el sistema MS-DOS utilizaba el llamado código 437 que incluía símbolos y caracteres de otras lenguas de Europa Occidental y caracteres que permitían hacer marcos y bordes en pantallas de texto , entre otros símbolos. Sin embargo 8 bits siguen siendo insuficientes para codificar todos los alfabetos del planeta. Por lo que cada zona usaba su propia tabla ASCII extendida. Ante el caos consiguiente, la ISO decidió normalizar dichas tablas de códigos para conseguir versiones estándares de los

mismos. Lo hizo mediante las siguientes normas (cada una de las cuales definía una tabla de 256 caracteres, siempre los 128 primeros son el ASCII original)

- 8859-1. ASCII extendido para Europa Occidental (incluye símbolos como ñ o β)
- 8859-2. ASCII extendido para Europa Central y del Este (incluye símbolos como Ź o č)
- 8859-3. ASCII extendido para Europa del Sur (incluye símbolos como Ġ o Ï)
- 8859-4. ASCII extendido para Europa del Norte (incluye símbolos como ø o å)
- 8859-5. ASCII extendido para alfabeto cirílico (incluye símbolos como д o Ж)
- 8859-6. ASCII extendido para alfabeto árabe (incluye símbolos como ن o ن)
- 8859-7. ASCII extendido para alfabeto griego moderno (incluye símbolos como φ o α)
- 8859-8. ASCII extendido para alfabeto hebreo (incluye símbolos como ק o ץ)
- 8859-9. ASCII extendido, versión de 8859-1 que incluye símbolos turcos en lugar de otros poco utilizados
- 8859-10. ASCII extendido, versión de 8859-4 que incluye símbolos más utilizados en las lenguas nórdicas actuales
- 8859-11. ASCII extendido para alfabeto tailandés (incluye símbolos como ฬ o ฮ)
- 8859-12. ASCII extendido para alfabeto devanagari de India y Nepal que ya no se usa
- 8859-13. ASCII extendido para alfabetos bálticos con símbolos que no estaban en 8859-4
- 8859-14. ASCII extendido para alfabeto celta (incluye símbolos como ŵ o Ẃ)
- 8859-15. ASCII extendido, versión de 8859 -1 que incluye el símbolo del euro y símbolos de lenguas bálticas. Es el recomendado actualmente para Europa Occidental.
- 8859-16. ASCII extendido, versión de 8859-1 pensada para los países del sureste de Europa
- 2022-JP. Símbolos japoneses (parte 1)
- 2022-JP-2. Símbolos japoneses (parte 2)
- 2022-KR. Símbolos coreanos

Este problema sigue existiendo ahora de modo que en los documentos de texto hay que indicar el sistema de codificación utilizado (el caso más evidente son las páginas web), para saber cómo interpretar los códigos del archivo. Así en 8859_1 el código 245 es el carácter õ y en 8859_2 es el carácter ő

Unicode

La complicación de las tablas de código se intenta resolver gracias al sistema Unicode que ha conseguido incluir los caracteres de todas las lenguas del planeta a cambio de que cada carácter ocupe más de un byte (ocho bits). En Unicode a cada símbolo se le asigna un número (evidentemente los 128 primeros son los originales de ASCII para mantener la compatibilidad con los textos ya codificados y de hecho los 256 primeros son la tabla ISO-8859_1).

Para ello el organismo también llamado Unicode participado por numerosas e influyentes empresas informáticas y coordinado por la propia ISO, ha definido tres formas de codificar los caracteres:

- UTF- 8. Es la más utilizada (y la más compleja de usar para el ordenador). Utiliza para cada carácter de uno a cuatro Bytes, de forma que:
  - Utilizan uno los que pertenecen al código ASCII original
  - Dos los pertenecientes a lenguas latinas, cirílicas, griegas, árabes, hebreas y otras de Europa, Asia Menor y Egipto

  - Tres para símbolos fuera de los alfabetos anteriores como el chino o el japonés.
  - Cuatro para otros símbolos: por ejemplo los matemáticos y símbolos de lenguas muertas como el fenicio o el asirio o símbolos asiáticos de uso poco frecuente.
- UTF-16. Utiliza para cada carácter dos (para los dos primeros grupos del punto anterior) o cuatro caracteres (para el resto). Es más sencillo que el anterior.
- UTF-32. La más sencilla de todas. Cada carácter independientemente del grupo al que pertenezca ocupa 4 caracteres. No se utiliza.

1.3 Archivos binarios y archivos de texto

Ventajas de los archivos binarios

1. Ocupan menos espacio que los archivos de texto, ya que optimizan mejor su codificación a binario (por ejemplo el número 213 ocupa un solo byte y no tres como ocurriría si fuera un texto).
2. Son más rápidos de manipular por parte del ordenador (se parecen más al lenguaje nativo del ordenador)
3. Permiten el acceso directo a los datos. Los archivos de texto siempre se maneja de forma secuencial, más lenta
4. En cierto modo permiten cifrar el contenido que de otra forma sería totalmente visible por cualquier aplicación capaz de entender textos (como el bloc de notas). Es decir los datos no son fácilmente entendibles

Ventajas de los archivos de texto

1. Son ideales para almacenar datos para exportar e importar información a cualquier dispositivo electrónico ya que cualquier es capaz de interpretar texto
2. Son directamente modificables, sin tener que acudir a software específico
3. Su manipulación es más sencilla que la de los archivos binarios
4. Son directamente transportables y entendibles por todo tipo de redes

2. EXPORTAR / IMPORTAR DATOS

2.1 El problema de compartir datos

Los problemas relacionados con el intercambio de información entre aplicaciones y máquinas informáticas es tan viejo como la propia informática.

El problema parte del hecho de haber realizado un determinado trabajo con un software en un ordenador concreto y después querer pasar dicho trabajo a otro software en ese u otro ordenador. Los archivos binarios tienen la complicación de que para hacer ese proceso, el origen y

el destino de los datos deben comprender cómo codificar y decodificar la información.

Eso, en muchos casos, ha sido un gran problema que ha obligado a que todos los trabajadores y trabajadoras hayan tenido que adaptarse al software de la empresa y por supuesto en toda la empresa utilizar dicho software.

En la informática actual eso es aún más problema al tener una necesidad de disponibilidad global del trabajo y además la posibilidad de ver dicho trabajo en dispositivos de todo tipo como mini ordenadores, PDA o incluso teléfonos móviles. Por ello poco a poco han aparecido formatos binarios de archivo que han sido estándares de facto (aunque no han sido reconocidos por ningún organismo de estándares) como por ejemplo el formato documental PDF, el formato de imagen JPEG, la música MP3 o el formato MPEG de vídeo.

Pero sigue habiendo empresas que utilizan formato propio por la idea de que sus formatos de archivo están directamente relacionados con la calidad de su software es decir razonan que el software que fabrican es muy potente y necesitan un formato binario propio compatible con esa potencia. De ahí que muchas veces la opción para exportar e importar datos sea utilizar conversores, capaces de convertir los datos de un formato a otro (por ejemplo de Word a OpenOffice; de MP3 a MOV de Apple, etc.).

2.2 Texto como el formato más versátil

Sin embargo hay un formato de archivo que cualquier dispositivo es capaz de entender. El texto. La cuestión es que para los archivos llamados de texto, sólo son capaces de almacenar texto plano; es decir sólo texto sin indicar ningún formato o añadir información no textual.

Debido a la facilidad de ser leído con cualquier aparato, se intenta que el propio texto sirva para almacenar otros datos. Evidentemente no es posible usar texto para almacenar por ejemplo imágenes, pero sí otras cosas. Para ello dentro del archivo habrá contenido que no se interpretará como texto sin más que simplemente se debe mostrar, sino que hay texto en el archivo que se marca de manera especial haciendo que signifique otra cosa. Desde hace muchos años hay dos campos en los que está idea ha funcionado bien: en las bases de datos y en los procesadores de texto. Actualmente el éxito de Internet ha permitido espolear esta tecnología a otros campos.

Hay un problema con el texto, puesto que al ser formato tan universal, y ser su contenido siempre visible; es peligroso como fuente para almacenar datos confidenciales, ya que quedaría expuesto a cualquier persona.

Los datos binarios no son del todo seguros, pero como requieren del software que entienda el formato binario concreto hacen que su contenido quede menos expuesto.

3. EJEMPLOS DE USO DE LOS CONJUNTOS DE CARACTERES

A continuación se exponen algunos ejemplos de dónde podemos encontrarnos la utilización de los conjuntos de caracteres para poder hacerse una idea de su utilización.

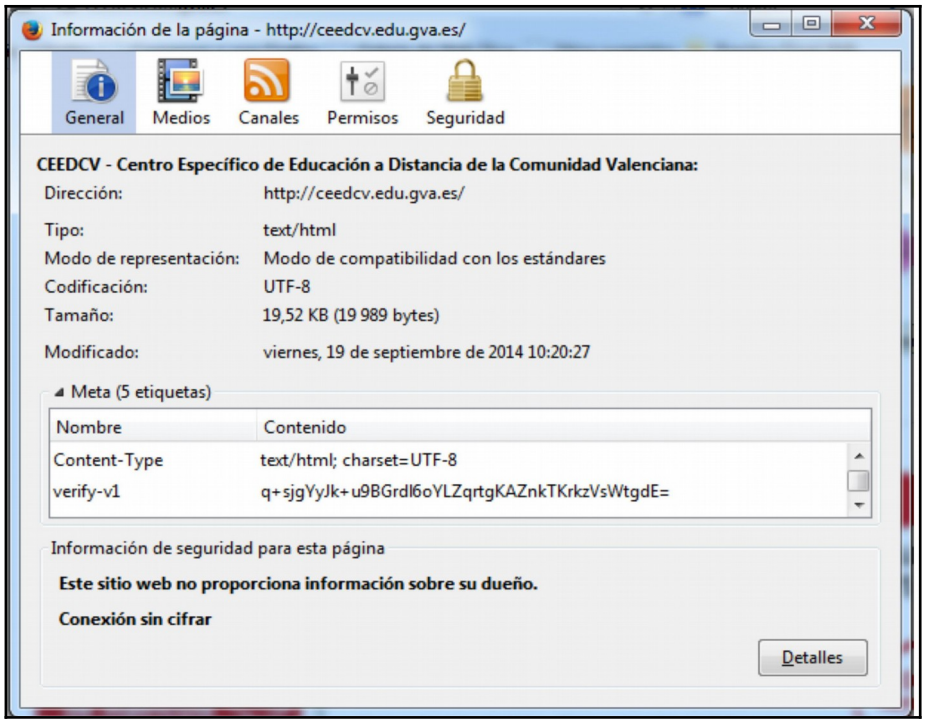
3.1 Páginas HTML

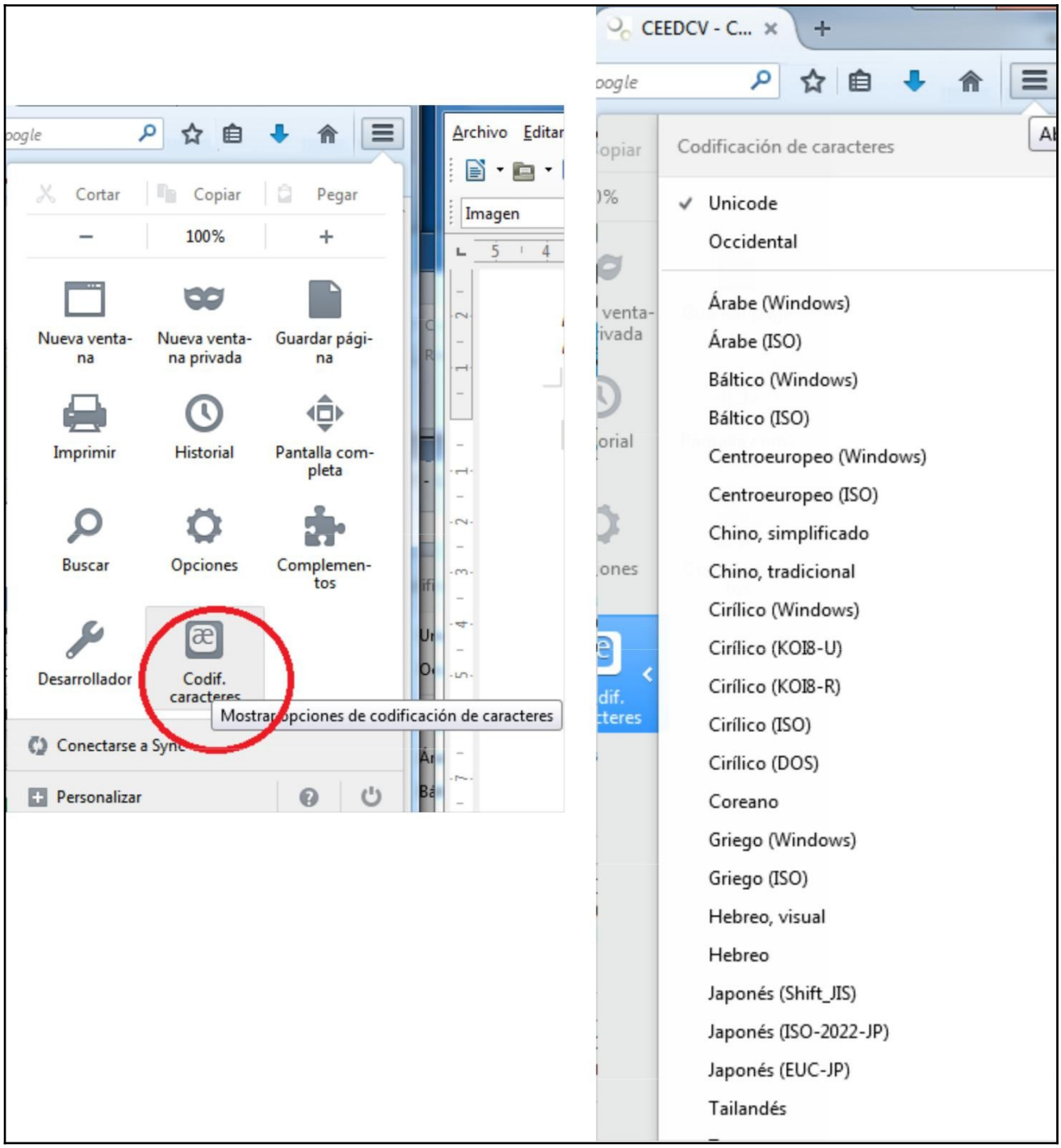
Cuando estamos diseñando una página web con código HTML tenemos que indicarle a través de las etiquetas <meta> el tipo de codificación de caracteres que vamos a utilizar dentro de ella, para que el navegador sepa interpretar los caracteres de manera correcta.

Unos ejemplos de etiquetas serían los siguientes:

```
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
```

El navegador leerá dicha información y se configurará automáticamente para mostrar el contenido utilizando dicho conjunto de caracteres. En Mozilla Firefox podemos consultar la información de la página y la codificación utilizada como observamos en las siguientes imágenes.





3.2 Bases de datos

Cuando creamos una base de datos debemos de elegir siempre el conjunto de caracteres que se va a utilizar para almacenar la información, pero además es muy importante también elegir el cotejamiento para seleccionar el tipo de ordenación y cómo trabajarán las búsquedas.

El cotejamiento (collation) tiene que ver con el orden que van a tener los caracteres o símbolos dentro de un conjunto de caracteres, y que no tiene que ver con el orden en el que se

almacenan dentro del conjunto.

Un sistema de cotejamiento puede ser la ordenación numérica, que fija un orden para los números positivos, negativos, decimales, etc. Otro sistema de cotejamiento puede ser la ordenación alfabética, que fija un orden para las letras del alfabeto. O incluso un sistema de cotejamiento para símbolos como en el caso del lenguaje chino.

El sistema de cotejamiento, a la hora de ordenar palabras, números, etc; indicará si ordenará teniendo en cuenta los caracteres de izquierda a derecha (como en español) o de derecha a izquierda (como en árabe).

Lo normal es que nos encontremos cotejamientos que sean no-sensibles a mayúsculas, minúsculas y acentos (case insensitive - ci) y, por tanto, ordenen al mismo nivel una letra en mayúscula, minúscula y con acento. Aunque también podemos encontrar lo contrario.

Además podemos encontrar que para un mismo conjunto de caracteres existan varios cotejamientos, normalmente según el idioma con el que trabajemos. Por ejemplo, UTF-8 da soporte a casi la totalidad de lenguas del mundo, por lo que existirá un cotejamiento diferente para cada idioma. El carácter Ñ tendrá su sitio solamente en el cotejamiento español (spanish).

3.3 Sistemas Operativos

Mientras que los sistemas operativos Linux y el sistema Operativo iOS de Apple utiliza el conjunto de caracteres UTF- 8 a la hora del nombre de archivos y directorios y del contenido de archivos de texto plano, los sistemas operativos Windows de Microsoft siguen utilizando el conjunto de caracteres ISO-8859, con la variante específica según el lugar donde se esté utilizando.

Estas diferencias harán que los ficheros creados en un sistema Windows muestren los caracteres de manera diferente en el resto de sistemas operativos, por lo que tendremos que tenerlo en cuenta.

4. LENGUAJES DE MARCAS

4.1 Introducción histórica

Aparición de los lenguajes de marcas

Como se ha comentado en el punto anterior, el problema de la exportación de datos ha puesto en entredicho a los archivos binarios como fuente para exportar e importar información. En su lugar parece que los archivos de texto poseen menos problemas (excepto el del cifrado de su información, que queda demasiado descubierta). Por ello se ha intentado que los archivos de texto plano (archivos que sólo contienen texto y no otros datos binarios) pudieran servir para almacenar otros datos como por ejemplo detalles sobre el formato del propio texto u otras indicaciones.

Los procesadores de texto fueron el primer software en encontrarse con este dilema. Puesto que son programas que sirven para escribir texto parecía que lo lógico era que sus datos se almacenaran como texto. Pero necesitan guardar datos referidos al formato del texto, tamaño de la página, márgenes, etc. La solución clásica ha sido guardar la información de formato de forma binaria, lo que provoca los ya comentados problemas. Algunos procesadores de texto optaron por guardar toda la información como texto, haciendo que las indicaciones de formato no se almacenen de forma binaria sino textual.

Dichas indicaciones son caracteres marcados de manera especial para que así un programa adecuado pueda traducir dichos caracteres no como texto sino como operaciones que finalmente producirán mostrar el texto del documento de forma adecuada. La idea del marcado procede del inglés marking up término con el que se referían a la técnica de marcar manuscritos con lápiz de color para hacer anotaciones como por ejemplo la tipografía a emplear en las imprentas. Este mismo término se ha utilizado para los documentos de texto que contienen comandos u anotaciones. Las posibles anotaciones o indicaciones incluidos en los documentos de texto han dado lugar a lenguajes (entendiendo que en realidad son formatos de documento y no lenguajes en el sentido de los lenguajes de programación de aplicaciones) llamados **lenguajes de marcas, lenguajes de marcado o lenguajes de etiquetas.**

Goldfarb

Se considera a Charles Goldfarb como al padre de los lenguajes de marcas. Se trata de un investigador de IBM que propuso ideas para que los documentos de texto tuvieran la posibilidad de indicar el formato del mismo. Al final ayudó a realizar el lenguaje GML de IBM el cual puso los cimientos del futuro SGML ideado por el propio Goldfarb.

TeX y LaTeX

En la década de los 70 Donald Knuth (uno de los ingenieros informáticos más importantes de la historia, padre del análisis de algoritmos) creó para producir documentos científicos utilizando una tipografía y capacidad es que fueran iguales en cualquier computadora, asegurando además una gran calidad en los resultados.

Para ello apoyó a TeX con tipografía especial (fuentes Modern Computer) y un lenguaje de definición de tipos (METAFONT).

TeX ha tenido cierto éxito en la comunidad científica gracias a sus 300 comandos que permiten crear documentos con tipos de gran calidad, para ello se necesita un programa capaz de convertir el archivo TeX a un formato de impresión.

El éxito de TeX produjo numerosos derivados de los cuales el más popular es (LaTeX). Se trata de un lenguaje que intenta simplificar a TeX, fue definido en 1984 por Leslie Lamport, aunque después ha sido numerosas veces revisado. Al utilizar comandos de TeX y toda su estructura tipográfica, adquirió rápidamente notoriedad y sigue siendo utilizado para producir documentos con expresiones científicas, de gran calidad. La idea es que los científicos se centren en el contenido y no en la presentación.

Ejemplo de código LaTeX:

```
\documentclass[12pt]{article}
\usepackage{amsmath}
\title{\Ejemplo}
\begin{document}
Este es el texto ejemplo de \LaTeX{}
Con datos en \emph{cursiva} o \textbf{negrita}.
Ejemplo de f\'ormula
\begin{align}
E &= mc^2
\end{align}
\end{document}
```

Que con un traductor daría lugar al siguiente resultado:

Este es el texto ejemplo de LaTeX
Con datos en *cursiva* o **negrita**. Ejemplo de fórmula

$$E = mc^2 \quad (1)$$

RTF

RTF es el acrónimo de Rich Text Format (Formato de Texto Enriquecido) un lenguaje ideado por Microsoft en 1987 para producir documentos de texto que incluyan anotaciones de formato.

Actualmente se trata de un formato aceptado como texto con formato y en ambiente Windows es muy utilizado como formato de intercambio entre distintos procesadores por su potencia. El procesador de texto Word Pad incorporado por Windows lo utiliza como formato nativo. Ejemplo:

```
{\rtf1\ansi\ansicpg1252\deff0\deflang3082{\fonttbl{\f0\fnil\fcharset0 Calibri;}}
\viewkind4\uc1\pard\sa200\sl276\slmult1\lang10\f0\fs22 soy \i cursiva\i0\par}
```

Produce el resultado:

soy cursiva

SGML

Se trata de la versión de GML que estandarizaba el lenguaje de marcado y que fue definida finalmente por ISO como estándar mundial en documentos de texto con etiquetas de marcado. La estandarización la hace el subcomité SC24 que forma parte del comité JTC1 del organismo IEC de ISO que se encarga de los estándares electrónicos e informáticos (en definitiva se trata de una norma ISO/IEC JTC1/SC2 4, concretamente la 8879).

Su importancia radica en que es el padre del lenguaje XML y la base sobre la que se sostiene el lenguaje HTML. En SGML las etiquetas que contienen indicaciones para el texto se colocan entre símbolos < y >.

Las etiquetas se cierran con el signo /. Es decir las reglas fundamentales de los lenguajes de etiquetas actuales ya las había definido SGML. En realidad (como XML) no es un lenguaje con unas etiquetas concretas, sino que se trata de un lenguaje que sirve para definir lenguajes de etiquetas; o más exactamente es un lenguaje de marcado que sirve para definir formatos de documentos de texto con marcas. Entre los formatos definidos mediante SGML, sin duda HTML es el más popular.

PostScript

Se trata de un lenguaje de descripción de páginas. De hecho es el más popular. Permite crear documentos en los que se dan indicaciones potentísimas sobre como mostrar información en el dispositivo final. Se inició su desarrollo en 1976 por John Warnocky dos años más tarde se continuo con la empresa Xerox, hasta que en 1985 el propio Warnock funda Adobe Systems y desde esa empresa se continua su desarrollo.

Es en realidad todo un lenguaje de programación que indica la forma en que se debe mostrar la información que puede incluir texto y el tipo de letra del mismo, píxeles individuales y formas vectoriales (líneas, curvas). Sus posibilidades son muy amplias.

Ejemplo:

```
%!PS
/Courier        % Elige el tipo de letra
20 selectfont   % Establece el tamaño de la letra y
                % la toma como el tipo de letra en uso
72 500 moveto           % Coloca el cursor en las coordenadas
                % 72, 500 (contando los píxeles desde
                la esquina izquierda de la página)
(Hola mundo!) show % Escribe el texto entre paréntesis,
showpage % Imprime el resultado
```

HTML

Tim Bernes Lee utilizó SGML para definir un nuevo lenguaje de etiquetas que llamó Hypertext Markup Language (lenguaje de marcado de hipertexto) para crear documentos transportables a través de Internet en los que fuera posible el hipertexto; es decir, la posibilidad que determinadas palabras marcadas de forma especial permitieran abrir un documento relacionado con ellas.

A pesar de tardar en ser aceptado, HTML fue un éxito rotundo y la causa indudable del éxito de Internet. Hoy en día casi todo en Internet se ve a través de documentos HTML, que popularmente se denominan páginas web.

Inicialmente estos documentos se veían con ayuda de intérpretes de texto (como por ejemplo el Lynx de Unix) que simplemente coloreaban el texto y remarcaban el hipertexto. Después el software se mejoró y aparecieron navegadores con capacidad más gráfica para mostrar formatos más avanzados y visuales.

XML

Se trata de un subconjunto de SGML ideado para mejorar el propio SGML y con él definir lenguajes de marcado con sintaxis más estricta, pero más entendibles. Su popularidad le ha convertido en el lenguaje de marcado más importante de la actualidad y en el formato de documentos para exportación e importación más exitoso.

JSON

Abreviatura de JavaScript Object Notation, Se trata de una notación de datos procedente del lenguaje JavaScript estándar (concretamente ECMA Script de 1999). En el año 2002 se le daba soporte desde muchos de los navegadores y su fama ha sido tal que ahora se ha convertido en una notación independiente de JavaScript que compite claramente con XML.

Se trata de una notación que realmente no se considera lenguaje de marcas, ya que no hay diferencia en el texto a través de etiquetas, sino que se basa en que el texto se divide en dato y metadato. De modo que el símbolo de los dos puntos separa el metadato del dato. Por otro lado los símbolos de llave y corchete permiten agrupar de manera correcta los datos.

Ejemplo:

```
{
"nombre": "Jorge",
"apellido1": "Sánchez",
"dirección":
{
        "calle": "C/ Falsa nº 0",
        "localidad": "Palencia",
        "código Postal": "34001",
        "país":"España"
},
"teléfonos": [
        {
                "tipo": "fijo",
                "número": "999 999 999" },
        {
                "tipo": "móvil",
                "number": "666 666 666"
        }
]
}
```

4.2 Tipos de lenguajes de marcas

Orientados a la presentación . En ellos al texto común se añaden palabras encerradas en símbolos especiales que contienen indicaciones de formato que permiten a los traductores de este tipo de documentos generar un documento final en el que el texto aparece con el formato indicado. Es el caso de **HTML** en el que se indica cómo debe presentarse el texto (y no por ejemplo lo que significa el mismo) también se considera así los archivos generados por los procesadores de texto tradicionales en los que al texto del documento se le acompaña de indicaciones de formato (como negrita, cursiva, etc.).

Orientados a la descripción. En ellos las marcas especiales permiten dar significado al texto pero no indican cómo se debe presentar en pantalla el mismo. Sería el caso de **XML** (o de SGML) y **JSON** en el que la presentación nunca se indica en el documento; simplemente se indica una semántica de contenido que lo hace ideal para almacenar datos (por ejemplo si el texto es un nombre de persona o un número de identificación fiscal).

Orientados a procedimientos. Se trata de documentos en los que hay texto marcado especialmente que en realidad se interpreta como órdenes a seguir y así el archivo en realidad contiene instrucciones a realizar con el texto. Es el caso de LaTeX o PostScript donde por ejemplo se puede indicar una fórmula matemática.