IES ANTONIO SEQUEROS

Financiado por la Unión Europea NextGenerationEU

Lenguajes de marcas y sistemas de gestión de información.

UT 1. Introducción a o lenguajes de marcas.

1. Ficha pedagógica.

Temporalización: 10 horas.	
RESULTADOS DE APRENDIZAJE	RA1
OBJETIVOS GENERALES	h,p
COMPETENCIAS	o, p, w
CONTENIDOS	
Introducción. Lenguajes. Gestión y representación de la información. Introducción a los lenguajes de marcas. Concepto de lenguaje de marcas y características. Tipos de lenguajes de marcas. Ventajas. Necesidad de uso. XML. Etiquetas. Estructura. Caracteres especiales. Documentos XML bien formados Lenguajes y ámbitos de aplicación. Clasificación Herramientas de edición. Otros formatos de texto estructurados.	
CRITERIO DE EVALUACIÓN	a) Se han identificado las características generales de los lenguajes de marcas. b) Se han reconocido las ventajas que proporcionan en el tratamiento de la información.

	c) Se han clasificado los lenguajes de marcas e identificado los más relevantes. d) Se han diferenciado sus ámbitos de aplicación. e) Se han reconocido la necesidad y los ámbitos específicos de aplicación de un lenguaje de marcas de propósito general. f) Se han analizado las características propias de diferentes lenguajes de marcas. g) Se han identificado la estructura de un documento y sus reglas sintácticas. h) Se han contrastado la necesidad de crear documentos bien formados y la influencia en su procesamiento. i) Se han identificado las ventajas que aportan los espacios de nombres.

2. Contenidos.

2.1. Introducción.

2.1.1. Lenguajes.

Los lenguajes en general y los lenguajes usados en informática han de definir:

- Léxico: Define las palabras que existen en el lenguaje, en el mundo informático se conocen como “tokens”, por ejemplo, tomate es una palabra válida en castellano, pero otamet no está definido.
- Sintaxis: Se establen las reglas que permiten a partir de las palabras construir expresiones, por ejemplo, un lenguaje matemático sencillo posee el léxico siguiente: +, -, * ,/, =, [0-9]+, donde [0-9] define números del 0 al 9 y + que ha de existir al menos 1.

  La expresión 1++=2/=8 es léxicamente correcta en el lenguaje, pero no sintácticamente

- Semántica: Existen ciertas condiciones que no es posible expresar con reglas sintácticas, en especial el significado y la lógica de las extresiones. El ejemplo clásico es intentar realizar operaciones con elementos que no tiene lógica que se realicen como puede ser sumar una letra a un número: a+45.

La frase:

El gato verde canta en la cima del árbol mientras las nubes bailan alrededor.

¿Es correcta desde el punto de vista léxico, sintáctico y semántico?

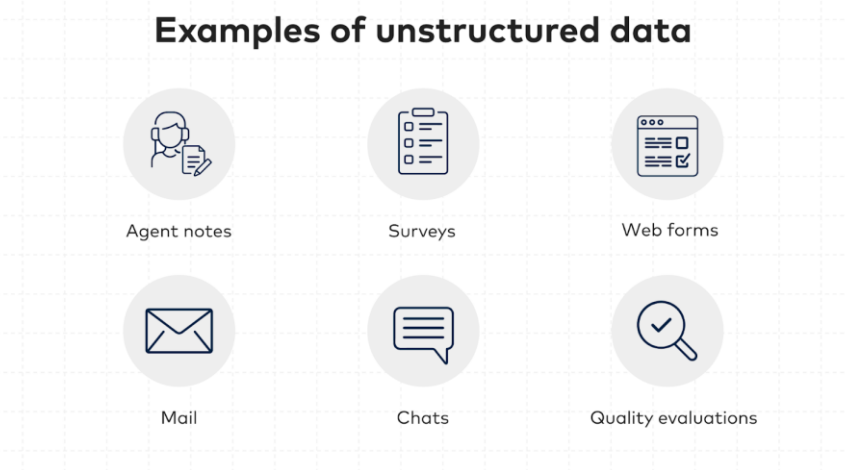
Los lenguajes de marcas también poseen un léxico, una sintaxis y una semántica.

2.1.2. Gestión y representación de información.

El almacenamiento, gestión e intercambio de información de información ha sido, y sigue siendo una de las tareas más destacadas de la informática. La información se puede clasificar principalmente en 2 tipos:

Información no estructurada. La información no posee una estructura común, no se repite entre elementos y no se puede garantizar la existencia de datos en esta información. No sigue un modelo predefinido, por ejemplo, documentos de texto o publicación en redes sociales. Sus características son:

- Difícil de organizar y buscar: Debido a la falta de un esquema predefinido, estos datos no se pueden consultar ni organizar fácilmente.
- Alta variabilidad: Los datos pueden presentarse en cualquier forma y formato.
- Procesamiento complejo: Requiere tecnologías avanzadas como el procesamiento del lenguaje natural (NLP) o algoritmos de machine learning para su análisis.



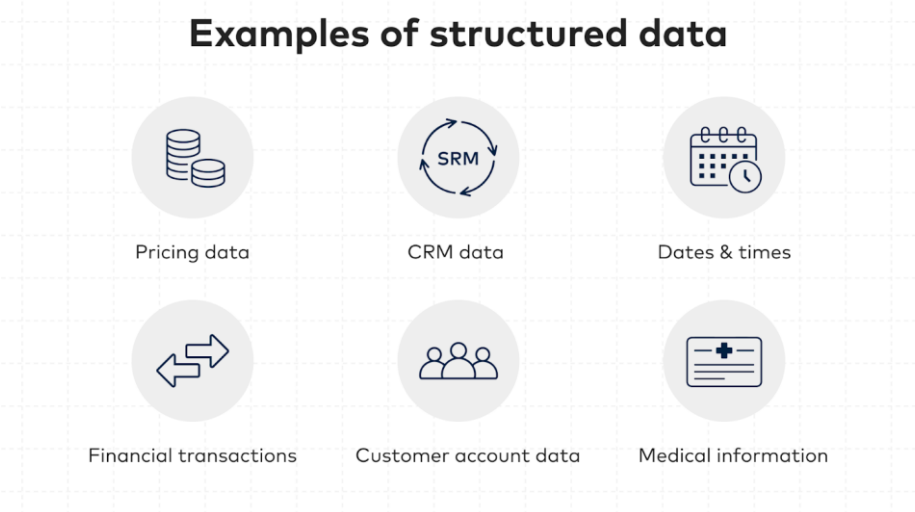
Información estructurada. Posee como su nombre indica una estructura común entre los elementos que la componen, de forma que es posible tratar la información contenida. Sus características son:

- Fácil de organizar y buscar: Dado que sigue un esquema predefinido, se puede organizar y consultar eficientemente.
- Alta coherencia: Todos los datos siguen un formato estándar.
- Facilidad de procesamiento: Las máquinas pueden procesar y analizar estos datos de manera eficiente usando algoritmos estándar.

Si bien la información estructurada ha sido la más común en el mundo de la informática por facilidad de procesamiento, en los últimos años el tratamiento de información no estructurada, especialmente con gran cantidad de datos ha experimentado una evolución considerable.

En este módulo se trabaja con principalmente con XML, un lenguaje de definición de lenguajes que representan información estructurada, en especial HTML, junto con diferentes tecnologías desarrolladas para trabajar con este lenguaje como son la corrección de la información, la búsqueda o la transformación.

XML no es la única tecnología utilizada para la representación estructurada, existiendo otras, aunque no tan completas como XML que se utilizan con frecuencia, en especial JSON y en menor medida YAML, tratándose también en este módulo.



2.2. Introducción a los lenguajes de marcas.

La informática tiene como uno de sus principales fines la representación, tratamiento y transferencia de información. Los primeros programas ya almacenaban de una u otra forma información, aunque cada uno con un formato propio que hacía muy complicada la transferencia de información, además el formato no ofrecía información extra necesaria para su comprensión por terceros (humanos u otros sistemas). Por ejemplo, un fichero con siguientes datos:

25-26-16-45-(-23)

¿Qué significan cada uno de esos números? Si se envía a otro sistema ¿Cómo saber si es correcto enviar 5 números o 6 números? ¿-23 es un valor válido?

Los lenguajes de marcas resuelven los anteriores problemas y algunos otros añadiendo información a los datos, conocidos por metainformación (definición: es la

información que describe, contextualiza o caracteriza otros datos. Proporciona detalles sobre los datos principales, facilitando su organización, comprensión, búsqueda y uso).

Se añaden a los datos en bruto elementos que los estructuras y describen, aportando por tanto información. Siguiendo con el ejemplo anterior:

Cotización:

Lunes: 25

Martes: 26

Miercoles: 16

Jueves: 45

Viernes: -23

Se han añadido metainformación a los datos, usando un formato estructurado usando los dos puntos:, el tabulado y el salto de línea.

2.2.1. Concepto de lenguaje de marcas y características.

Se define lenguaje de marcas como un sistema de codificación que añade etiquetas o “marcas” para definir y estructurar datos. Las etiquetas o marcas permiten añadir información a los datos, analizarlos e interpretarlos.

Las características de los lenguajes de marcas son:

Uso de etiquetas: Conjunto de caracteres que dan un significado a los datos contenidos en la etiqueta, en el caso de XML las etiquetas se definen usando los símbolos <, > y /, junto con el nombre que añade metainformación a los datos, en el caso de JSON (**no son exactamente etiquetas**), se utilizan [,],{,} o :.

Estructura jerárquica: Trabajan con información estructurada, y dentro de esta con información jerárquica, por ejemplo, una factura es información jerárquica, ya que esta posee los “hijos”: Estado, datos cliente, datos empresa, productos, y estos a su vez más hijos:

Factura:

- Estado.
- Datos del cliente:
  - Nombre.
  - Apellidos.
  - Dirección.
  - Teléfono.

Datos empresa:

Nombre:

Dirección.

CIF

Productos:

Línea:

Producto.

Unidades.

Precio.

Iva.

Legible por los humanos. Los lenguajes de marcas, además de ser procesados por máquinas, también pueden ser interpretados, creados y modificados por los humanos.

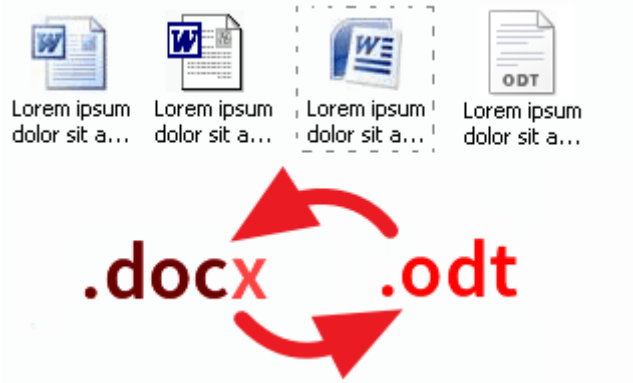
Se centra en los datos y el significado. No añaden información que no sean los propios datos y el significado de estos, por ejemplo, formatos de presentación o impresión.

2.2.2. Tipos de lenguajes de marcas.

Los lenguajes de marcas se pueden clasificar en función de diferentes criterios siendo el más común el de su propósito, aunque muchos de ellos se utilizan en diferentes propósitos. Se tiene lenguajes de marcas pensados para:

- **Presentación de documentos**. Utilizados para estructurar y dar formato a documentos. Por ejemplo, los documentos docx utilizan un lenguaje de marcas internamente.

- **De datos.** Su función es la de describir, transportar y almacenar datos de manera estructurada. Son los más utilizados en la actualidad.
- **De procesamiento.** Facilitar el procesamiento automatizado de información, por ejemplo: la transformación o filtrado como puede ser transformar de docx a odt.



- **De contenido multimedia.** Sirven para describir y gestionar contenido multimedia.
- **Ligeros.** Se centran en un formato sencillo que facilita el procesamiento y la transmisión.

2.2.3. Ventajas. Necesidad de uso.

Ya se ha mencionado algunos de los problemas que resuelven los lenguajes de marcas, de forma general: la creación, organización, intercambio y en menor medida la presentación de la información. Con más detalle las ventajas que ofrecen los lenguajes de marcas son:

- **Estructurar la Información**. Organizar información de forma estructurada, añadiendo metainformación sobre los datos.
- **Interoperabilidad**. Facilitar el intercambio de información entre diferentes sistemas.
- **Presentación y formateo de contenidos.** Si bien no debe de incluirse la presentación y el formateo de contenidos en los lenguajes de marcas (se tiende a ello desde hace unos años) aún existen lenguajes de marcas con etiquetas que añaden información sobre la presentación.
- **Automatización y procesamiento**. Facilitan la automatización y tratamiento de la información al poseer conocimiento sobre los datos y su significado, por ejemplo, la transferencia entre entidades bancarias.
- **Versatilidad y Extensibilidad.** Creación de nuevas etiquetas que creen o amplíen lenguajes existentes.
- **Validación.** Asegurar que los documentos y datos cumplen con las normas y especificaciones establecidas.
- **Independencia de la plataforma, sistema o dispositivo.** Permite que la información sea compartida e interpretada sin importar el sistema operativo o software.

2.3. XML.

XML es el lenguaje de marcas más conocido y completo, siendo un acrónimo de eXtensible Markup Language, traducido como 'Lenguaje de Marcado Extensible' o 'Lenguaje de Marcas Extensible'. Se define como un metalenguaje, un **lenguaje que permite definir otros lenguajes (la gramática).**

Es un estándar en el mundo de la informática utilizándose en un gran número de áreas. Su origen proviene de SGML, lenguaje de marcas definido por IBM en los años 60, SGML poseía una gran potencia y flexibilidad, pero estas características también implicaban gran capacidad de procesamiento y la implementación de toda la definición era complicada y difícil de llevar a cabo.

En 1991 se produjo uno de los eventos más destacados en la historia de la informática, Tim Berners-Lee, define en el CERT para el intercambio de información entre científicos el lenguaje HTML basándose en SGML. El problema era que asegurarse que las páginas web cumplieran con todas las reglas del lenguaje HTML con el estándar SGML no era posible por la capacidad de cómputo de los equipos y de los programas en los que se mostraban “renderizado”, los navegadores. Los analizadores de los navegadores eran muy flexibles con la corrección o no de la estructura de las páginas web.

A raíz del éxito de HTML la organización W3C crea en 1996 un grupo de trabajo para simplificar la complejidad de SGML, a esta simplificación se le denomino XML, apareciendo la primera versión en 1998 (actualmente se encuentra en la versión 1.1). Las características de esta adaptación son:

- Posibilidad de definición de lenguajes a partir del metalenguaje (XML).
- Creación de analizadores (ver si se ha escrito gramática y sintéticamente de forma correcta un documento) simples, necesitando por tanto poca capacidad de cómputo (comparado con SGML).
- Lenguajes para realizar búsquedas y transformaciones.

2.3.1. Etiquetas.

El elemento básico para la construcción de documentos XML es la etiqueta o “tag”, una etiqueta se puede definir léxicamente de forma general de la siguiente forma:

Inicio de etiqueta:

- Símbolo <.
- Nombre de la etiqueta
- Símbolo >

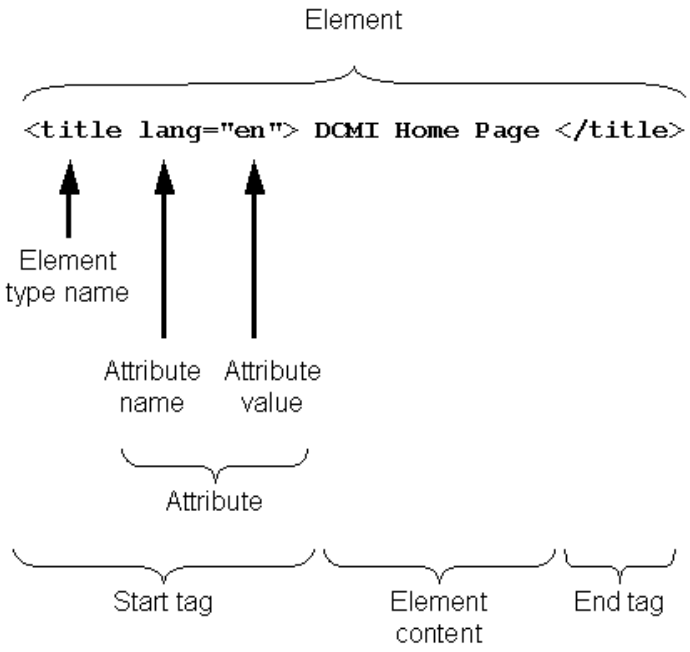
Contenido

Cierre de etiqueta:

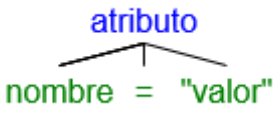
- Símbolo <.
- Símbolo /.
- Nombre etiqueta
- Símbolo >

Los nombres de las etiquetas han de cumplir algunas reglas:

- No pueden comenzar con un número o un carácter especial (excepto un guion bajo o dos puntos).
- No deben contener espacios.
- No pueden usar palabras reservadas de XML, como xml.



Si bien el nombre de la etiqueta añade metainformación a los datos, este puede que no sea suficiente, para añadir información personalizada a una etiqueta concreta se dispone de los atributos, que proporcionan información adicional sobre el elemento. Los atributos se definen en la apertura de la etiqueta, su léxico es:



Una etiqueta puede que no tenga atributo o que tenga varios:

```
<etiqueta nombre_atributo=”valor1” nombre_atributo2=”valor2”>...
```

Existen también etiquetas que no poseen contenido, denominadas etiquetas vacías, su sintaxis es:

```
<etiqueta/>
```

En la que se cierra y abre la etiqueta al mismo tiempo.

Algunos ejemplos de etiquetas reales:

- Usada en la definición de fórmulas matemáticas.

```
<mi>x</mi>
```

- Para imágenes vectoriales.

```
<circle class="circle" cx="5" cy="5" r="5" />
```

- Etiqueta usada en la factura electrónica.

```
<TaxIdentificationNumber>A2800056F</TaxIdentificationNumb
er>
```

- Etiqueta para definir un área de dibujo en una página web.

```
<canvas id="example" width="260" height="200"></canvas>
```

2.3.2. Estructura.

Una de las características más destacadas de XML es que el contenido de una etiqueta puede ser:

- Texto.
- Otra etiqueta.
- Conjunto de etiquetas.
- Vacío.

Esto implica que el documento XML se puede representar con una estructura **recursiva en forma de árbol.** A cada etiqueta se le denomina nodo en el árbol, el contenido si es texto también se considera un nodo. De cada nodo **que no sea texto** pueden colgar otros nodos, es decir etiquetas que contienen otras etiquetas.

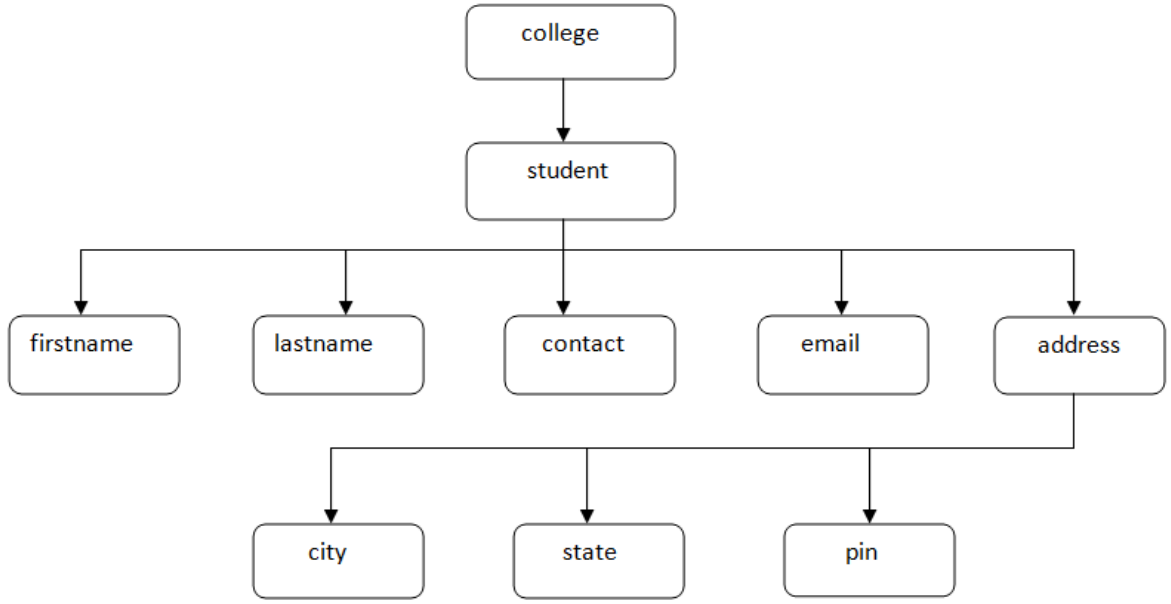
Una de las limitaciones de XML es que solo puede existir un nodo en el primer nivel, denominado raíz.

Por ejemplo, el siguiente XML:

```
<college>
  <student>
      <firstname>Tamanna</firstname>
      <lastname>Bhatia</lastname>
      <contact>09990449935</contact>
      <email>tammanabhatia@abc.com</email>
      <address>
          <city>Ghaziabad</city>
```

```
            <state>Uttar Pradesh</state>
            <pin>201007</pin>
        </address>
    </student>
</college>
```

La estructura en forma gráfica:



A falta de representar los nodos de texto. Se observa cómo solo existe un elemento o etiqueta raíz: ”collage”

Además, en un documento XML se han de incluir algún elemento más referida a la versión de XML en que se encuentra escrito el documento, que ha de ser el inicio de todo documento XML. El formato es:

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

Se indica la versión y de forma opcional el conjunto de caracteres que puede contener el documento normalmente es UTF-8 y si va acompañado de un conjunto de reglas con standalone.

También se puede incluir referencias a etiquetas válidas y sintaxis correcta para el contenido del documento, por ejemplo, si se un documento de factura electrónica no se podrán tener etiquetas de tipo <circulo> o no podrá aparecer el importe de la factura en los datos del cliente. **Al conjunto de etiquetas definidas en un ámbito se le denomina espacio de nombres.** En unidades posteriores se tratará con mayor profundidad, pero existen dos formas:

DTD. En el caso de los esquemas se utiliza doctype.

Para documentos SVG:

```
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.0//EN"
```

```
http://www.w3.org/TR/2001/REC-SVG-20010904/DTD/svg10.dtd>
```

En el caso de MathML:

```
<!DOCTYPE math SYSTEM "http://www.w3.org/TR/MathML2/dtd/mathml2.dtd">
```

Para páginas web:

HTML4.01, estricto:

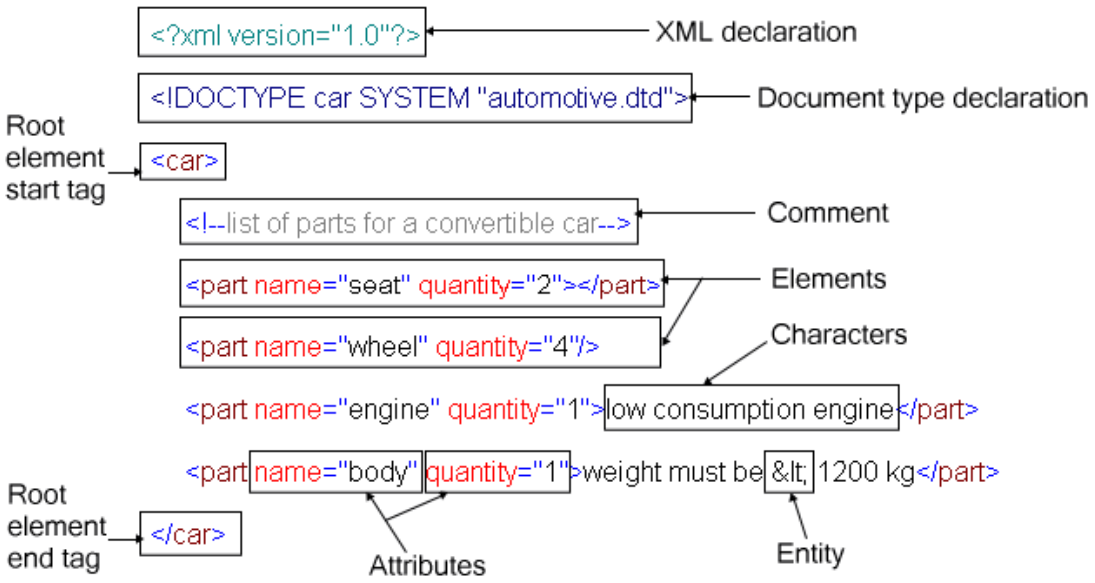
```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
```

HTML 5 se ha simplificado:

```
<!DOCTYPE html>
```

Schemas. En los esquemas se hace referencia como atributo en la raíz del documento. Se tratará en temas posteriores, un ejemplo simple:

```
<svg xmlns="http://www.w3.org/2000/svg" viewBox="0 0 100 100">
```



Es posible definir comentarios, que no afectaran al procesamiento del documento, los comentarios se definen con la notación:

```
<!-- Esto es un comentario -->
```

2.3.3. Caracteres especiales.

En la especificación de XML se utilizan caracteres que si se usan en el contenido o como nombre de los atributos crean ambigüedad y hace que los analizadores devuelven error, por ejemplo ¿qué sucede si el valor de un atributo contiene el símbolo >?.

Para solucionar este problema, se definen una serie de caracteres especiales y código asociados para su inclusión en un documento XML.

&lt;	<	less than
&gt;	>	greater than
&amp;	&	ampersand
&apos;	'	apostrophe
&quot;	"	quotation mark

2.3.4. Documentos XML bien formados.

En XML existen dos conceptos muy importantes sobre los documentos XML:

Documentos bien formados. Todo documento XML ha de estar bien formado, los analizadores lanzan errores en el caso de que un documento no cumpla con las **reglas comunes a todo XML,** independientemente del contenido o ámbito de uso.

Documentos válidos. Además de ser un documento bien formado, cumple con una serie de requisitos, normalmente relacionados con el ámbito de uso, **define un lenguaje para un ámbito particular**, por ejemplo, imágenes vectoriales, voz IP, intercambio de datos bancarios o cualquier otro que se desee.

Los requisitos para que un documento XML se encuentre bien formado son:

Toda etiqueta que se abre se ha de cerrar.

La primera etiqueta que se abre es la última en cerrarse.

Sólo existe un elemento raíz.

Se diferencia entre mayúsculas y minúsculas.

La forma correcta de cerrar etiquetas vacias es <etiqueta/>

El valor de los atributos han de encontrarse dentro de comillas dobles o simples.

Un ejemplo de XML mal formado:

```
<biblioteca id=1>
  <libro pages=”192>
    <titulo>El Quijote</titulo>
    <autor>Miguel de Cervantes</autor>
  </biblioteca>
```

```
<biblioteca id=”2”>
<libro pages=”192>
    <titulo>La Regenta</titulo>
    <autor>Leopoldo Alas</autor>
<Biblioteca/>
```

Un ejemplo de XML bien formado:

```
<svg   width="400" height="180">
  <!-- Rectángulo azul con bordes redondeados -->
  <rect x="50" y="20" rx="20" ry="20" width="150" height="100"
        style="fill:blue;stroke:black;stroke-width:5;opacity:0.7" />
  <!-- Círculo rojo -->
  <circle cx="250" cy="70" r="50"
          style="fill:red;stroke:black;stroke-width:5;opacity:0.7" />
  <!-- Línea diagonal verde -->
  <line x1="0" y1="0" x2="400" y2="180"
        style="stroke:green;stroke-width:5" />
  <!-- Texto -->
  <text x="200" y="150" font-family="Verdana" font-size="20" fill="black">
    Ejemplo de SVG Bien Formado
  </text>
</svg>
```

2.4. Lenguajes y ámbitos de aplicación.

Ya se ha indicado que XML es un metalenguaje, un lenguaje de marcas que permite definir otros lenguajes de marcas, en concreto el léxico (palabras, en este caso etiquetas y atributos) existentes en el lenguaje, la sintaxis (reglas y principios que gobiernan la combinatoria de elementos del lenguaje) y semántica (restricciones sobre el significado que la sintaxis no puede definir) del mismo.

Es posible definir lenguajes propios, pero existen una serie de lenguajes ya especificados para prácticamente cualquier necesidad del software, que definen sus propias reglas léxicas, sintácticas y semánticas.

2.4.1. Clasificación.

Existen cientos de lenguajes definidos a partir de XML. Para definir un nuevo lenguaje a partir del metalenguaje XML se utilizan las etiquetas y estructura vistos en puntos anteriores y se añaden las reglas léxicas, semáticas y sintácticas con DTP o esquemas (este último también es un lenguaje XML). algunos de los más conocidos clasificados por ámbito de aplicación:

1. Lenguajes de Descripción de Datos

Estos lenguajes se utilizan principalmente para definir la estructura y el contenido de los datos. Definen la sintaxis y semántica de los otros lenguajes.

- XML Schema (XSD): Define la estructura y los tipos de datos de un documento XML, asegurando que los documentos sigan un formato específico.
- Document Type Definition (DTD): Un lenguaje más antiguo y menos expresivo que XSD para definir la estructura de documentos XML.
- RELAX NG: Un lenguaje de esquema más simple y flexible que XSD, utilizado para la validación de documentos XML.
- Schematron: Se utiliza para la validación basada en reglas de documentos XML, complementando otros esquemas como XSD y RELAX NG.

2. Lenguajes de Transformación

Estos lenguajes permiten transformar documentos XML en otros formatos, como HTML, otro XML, o texto plano.

- XSLT (Extensible Stylesheet Language Transformations): El lenguaje más común para transformar documentos XML. Es parte del conjunto de tecnologías XSL.
- XPath: Un lenguaje para seleccionar partes de un documento XML, usado dentro de XSLT, XQuery, y otros lenguajes.

3. Lenguajes de Consulta

Se utilizan para consultar y extraer información de documentos XML.

- XQuery: Un potente lenguaje de consulta para extraer y manipular datos de documentos XML, similar a SQL para bases de datos relacionales.
- XPath: Además de su uso en XSLT, también se usa de manera independiente para localizar partes específicas de un documento XML.

4. Lenguajes de Enlace

Estos lenguajes permiten crear enlaces dentro de documentos XML o entre diferentes documentos.

- XLink (XML Linking Language): Permite definir enlaces entre recursos, similar a los hipervínculos en HTML, pero con mayor flexibilidad y complejidad.
- XPointer: Se usa junto con XLink para apuntar a partes específicas de un documento XML.

5. Lenguajes de Presentación

Estos lenguajes están diseñados para describir cómo se deben presentar o visualizar los datos XML.

- XSL-FO (XSL Formatting Objects): Utilizado para describir la presentación de documentos XML, principalmente en formato de impresión como PDF.
- SVG (Scalable Vector Graphics): Lenguaje basado en XML para describir gráficos vectoriales bidimensionales.

6. Lenguajes de Descripción de Interfaces y Servicios

Se utilizan para describir interfaces de servicios web y otros servicios basados en XML.

- WSDL (Web Services Description Language): Define los servicios web, describiendo cómo se accede a ellos y qué operaciones están disponibles.
- SOAP (Simple Object Access Protocol): Protocolo para intercambiar mensajes estructurados en un entorno distribuido, basado en XML.
- UDDI (Universal Description, Discovery, and Integration): Un estándar para la publicación y el descubrimiento de servicios web, utilizando XML para describir los servicios.

7. Lenguajes de Metadatos

Estos lenguajes se usan para describir metadatos o información sobre datos.

- RDF (Resource Description Framework): Un marco basado en XML para representar metadatos y describir recursos en la web.
- Dublin Core: Un conjunto de metadatos estándar que puede ser representado en XML para describir recursos de información, como documentos electrónicos.

8. Lenguajes Específicos de Dominio

Estos lenguajes están diseñados para aplicaciones específicas y aprovechan la flexibilidad de XML para adaptarse a diferentes industrias.

- MathML: Utilizado para describir notación matemática, permitiendo la representación de fórmulas matemáticas en documentos XML.
- DocBook: Un estándar para la creación de documentación técnica, utilizado principalmente para manuales y libros técnicos.
- HL7 (Health Level 7): Un estándar para el intercambio de información en el sector de la salud, donde XML se utiliza para estructurar los mensajes.

9. Lenguajes de Intercambio de Datos

Estos lenguajes están diseñados para facilitar el intercambio de datos entre sistemas heterogéneos.

- RSS (Really Simple Syndication): Un formato basado en XML para la distribución de contenido web actualizado frecuentemente, como blogs o noticias.
- Atom: Similar a RSS, pero con más funcionalidades para la sindicación de contenido web.
- SOAP: Además de ser un protocolo, SOAP también es un lenguaje de intercambio de datos, utilizado en la comunicación entre aplicaciones web.

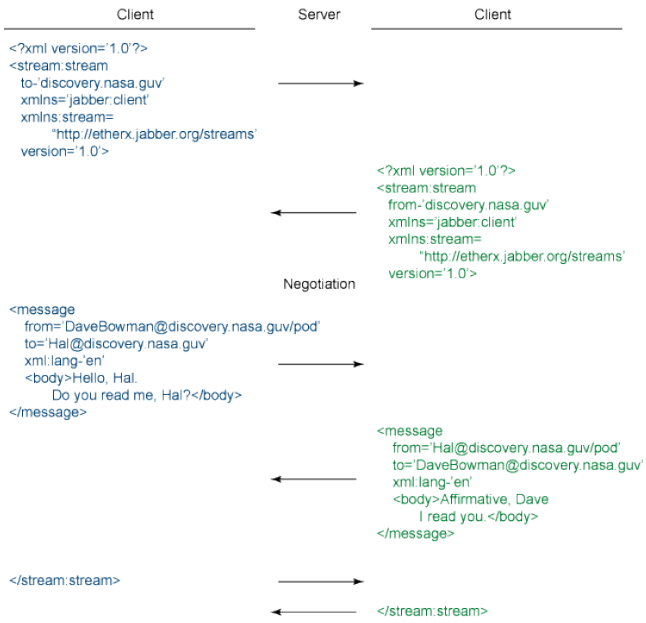
10. Lenguajes de Seguridad

Estos lenguajes se utilizan para añadir capas de seguridad en documentos XML.

- XML Signature: Permite la firma digital de documentos XML, asegurando la autenticidad e integridad del contenido.
- XML Encryption: Proporciona la capacidad de cifrar datos dentro de un documento XML, asegurando la confidencialidad.

2.5. Herramientas de edición.

Normalmente la información es consumida y generada por el software, por ejemplo, para enviar mensajes instantáneos con el protocolo XMPP,

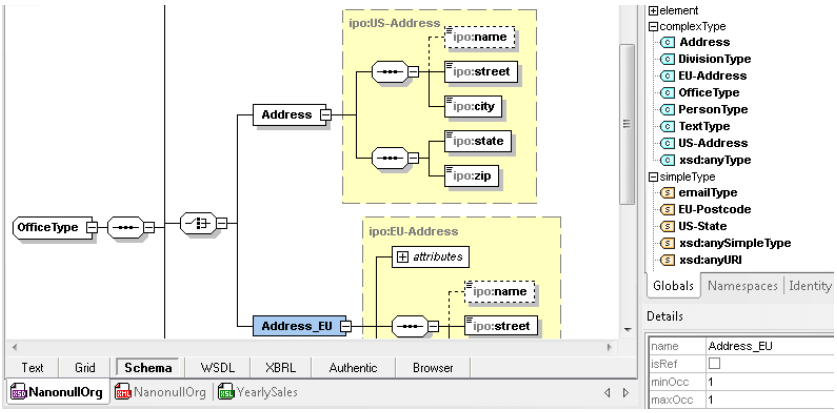


Pero en otras ocasiones es necesario editar de forma manual los ficheros XML: Definir esquemas, crear páginas web, crear transformaciones...

Con un simple editor de texto es suficiente para crear documentos XML como puede ser Notepad. El problema es que no asiste en la escritura del fichero, por ejemplo, errores que hacen que el documento no esté bien formado o se utilicen etiquetas que no.

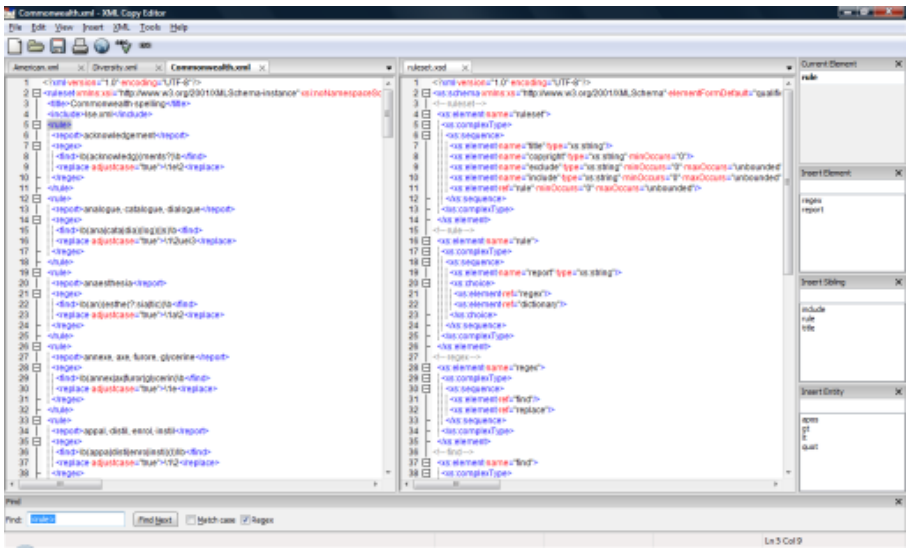
Existe software específico para trabajar con XML, los más conocidos son:

- Software de Oxigen. La empresa posee software especializado en todos los ámbitos relacionados con la edición, creación y definición de XML.
- XMLSpy. De la empresa Altova, permite trabajar también en todos los ámbitos de XML.

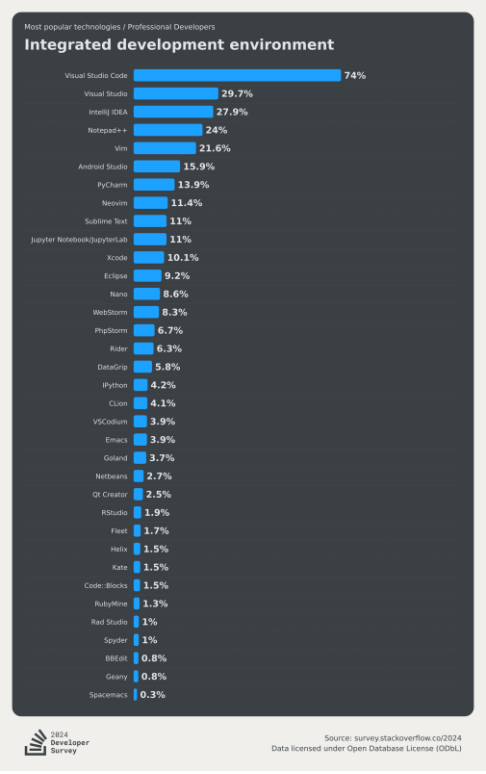


- XMLCopyEditor. Editor de software libre, multiplataforma y con un funcionamiento básico similar a los anteriores. Es ligero, rápido y ofrece una gama de herramientas útiles para la creación, edición y validación de documentos XML. Aunque no tiene tantas funcionalidades avanzadas como

algunos editores comerciales, es una excelente opción para quienes buscan una solución gratuita y eficiente para trabajar con XML.

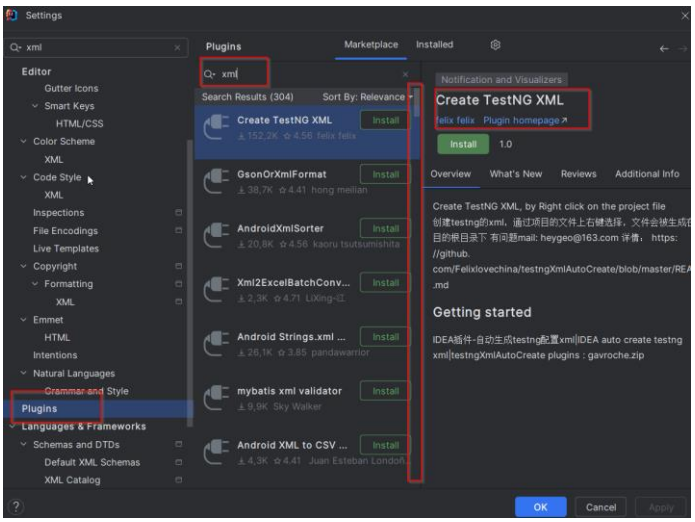


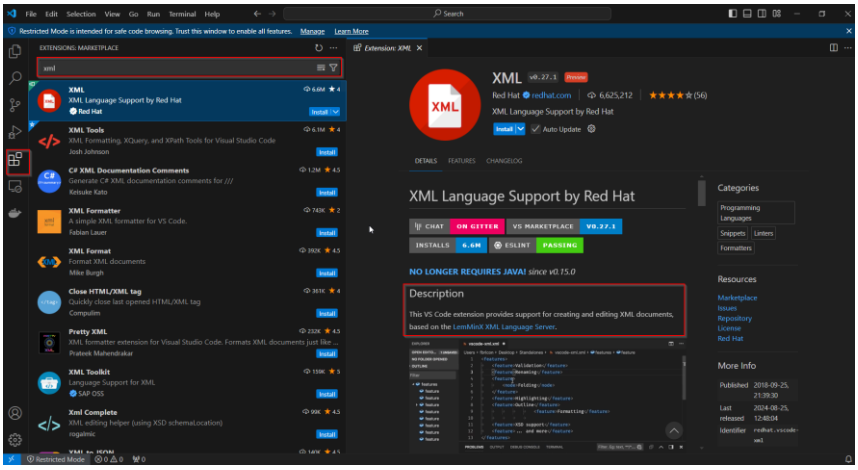
El software anterior es exclusivo para trabajar con XML, no siendo muy habitual que un desarrollador/a se dedique únicamente a XML sino que este sea un complemento en el desarrollo, utilizándose junto con algún lenguaje de programación y utilizando entornos de desarrollo también conocidos por IDE's. Prácticamente todos los IDE's con una cuota de uso significativa incluyen herramientas básicas para el manejo de XML.



Estadísticas encuesta Stackoverflow 2024.

Además, es posible añadir extensiones o “plugins” que ampliar las funcionalidades del IDE con respecto a XML, por ejemplo, en IntelliJ o VSC.





2.5.1. Espacios de nombres.

Los espacios de nombres en XML (XML namespaces) son una forma de evitar conflictos de nombres cuando se usan elementos o atributos de diferentes vocabularios XML en un mismo documento. Sirven para distinguir entre elementos o atributos que pueden tener el mismo nombre pero significados diferentes.

Cuando se combinan vocabularios o esquemas XML de distintas fuentes, pueden surgir conflictos de nombres. Por ejemplo, en un mismo documento puedes tener elementos llamados <title>, uno que pertenezca a un esquema que describe libros y otro que describa películas. Sin un mecanismo para distinguir entre estos usos, XML no podría diferenciarlos.

Un **espacio de nombres** es simplemente una URI (Uniform Resource Identifier), que actúa como identificador único para un conjunto de nombres. Aunque las URIs en espacios de nombres parecen URLs, no tienen que apuntar a un recurso específico en la web, solo actúan como identificadores.

2.5.1.1. Sintaxis de Espacios de Nombres en XML.

Declaración del Espacio de Nombres: Se declara en un elemento mediante el atributo xmlns seguido de un prefijo opcional o el xmlns por defecto.

- **Con Prefijo:** Se asocia un espacio de nombres con un prefijo que luego se utiliza como parte del nombre completo del elemento o atributo.

```
<bookstore xmlns:bk="http://www.ejemplo.com/libros">
    <bk:book>
        <bk:title>XML para principiantes</bk:title>
        <bk:author>Juan Pérez</bk:author>
    </bk:book>
</bookstore>
```

- **Espacio de nombres por defecto**: Si no deseas usar un prefijo, puedes declarar un espacio de nombres por defecto que se aplica a todos los elementos dentro de su alcance que no tienen un prefijo.

```
<bookstore xmlns="http://www.ejemplo.com/libros">
    <book>
        <title>XML para principiantes</title>
        <author>Juan Pérez</author>
    </book>
</bookstore>
```

Ejemplo con múltiples espacios de nombres.

Cuando se combinan diferentes vocabularios XML en un mismo documento, puedes declarar múltiples espacios de nombres con diferentes prefijos. Por ejemplo:

```
<store xmlns:bk="http://www.ejemplo.com/libros"
       xmlns:mv="http://www.ejemplo.com/peliculas">
    <bk:book>
        <bk:title>XML para expertos</bk:title>
        <bk:author>Maria Gómez</bk:author>
    </bk:book>
    <mv:movie>
        <mv:title>El mejor XML</mv:title>
        <mv:director>Pedro Fernández</mv:director>
    </mv:movie>
```

```
</store>
```

Los atributos también pueden estar en espacios de nombres, aunque no suelen usarse con tanta frecuencia. Un atributo pertenece al espacio de nombres si su nombre incluye un prefijo:

```
<bk:book bk:type="educativo">
    <bk:title>XML para principiantes</bk:title>
</bk:book>
```

2.6. Otros formatos de texto estructurados.

XML es el lenguaje de marcas muy extendido y con mayor funcionalidad de los mayoritarios, pero esta mayor funcionalidad implica también una mayor complejidad, además, de que en un alto porcentaje de situaciones esas funcionalidades no se utilizan. Por ejemplo, enviar un listado de productos a una aplicación en la que existen 1000 entradas, en envío y el procesamiento es muy costoso para el hardware si se compara con otros formatos.

Actualmente, tanto para el intercambio de datos entre sistemas, la definición de ficheros de configuración, o la solicitud de servicios a otros sistemas se utilizan además de XML otros formatos de texto (no son lenguajes), por supuesto**, no con la misma funcionalidad.** Los más utilizados son:

- **JSON**: JSON es un formato ligero para el intercambio de datos, fácil de leer y escribir para humanos y fácil de interpretar para máquinas. Es ampliamente utilizado en aplicaciones web y APIs debido a su simplicidad y compatibilidad con JavaScript.

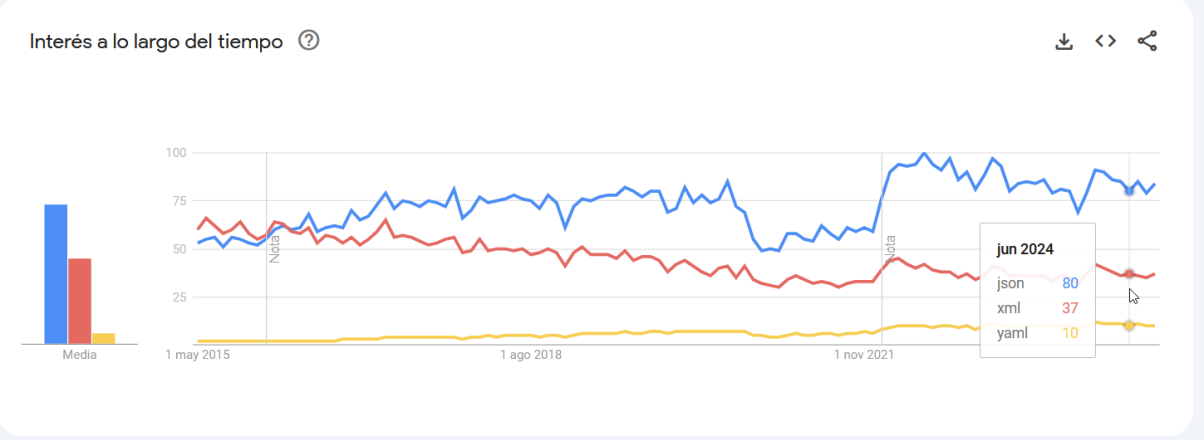
```
{
  "libro": {
    "titulo": "Don Quijote",
    "autor": "Miguel de Cervantes",
    "año": 1605
  }
}
```

- **YAML**: YAML es un lenguaje de serialización de datos legible para humanos, utilizado comúnmente para la configuración de aplicaciones y como un formato de intercambio de datos. Es más simple y menos verboso que XML o JSON, con una sintaxis más amigable.

```
libro:
  titulo: "Don Quijote"
```

```
autor: "Miguel de Cervantes"
año: 1605
```

- **CSV**: Simple y poco expresivo, se utiliza en el intercambio de datos tabulares. Cada línea representa una fila y los valores en las columnas están separados por comas (o algún otro delimitador).



Evolución de búsquedas en Google por lenguaje de marcas/formato de texto.

- Markdown: Es más un lenguaje de formato ligero que un lenguaje de marcas complejo, se utiliza ampliamente para dar formato a texto en una manera sencilla y legible. Se usa para escribir en plataformas como GitHub o en documentos que serán convertidos a HTML.
- LaTeX: Es un sistema de composición de textos que se utiliza principalmente en la creación de documentos científicos y técnicos. Utiliza comandos y etiquetas para definir la estructura de documentos complejos, como ecuaciones matemáticas.

2.6.1. JSON.

Si bien JSON no es propiamente un lenguaje de marcas, sino un lenguaje de intercambio de datos (o al menos lo fue inicialmente), su uso se ha extendido y se puede utilizar de forma similar a un lenguaje de marcas debido a su simplicidad y facilidad de procesamiento.

En JSON se tienen 3 elementos:

Conjuntos de entidades. Se indica con los símbolos **“[“** y **“]”**, dentro de cada conjunto se delimitan las entidades, o elementos primitivos (cadenas,números...) por el símbolo **“,”**. Puede contener diferentes tipos de entidades (aunque no suele ser lo habitual) y tantas como sean necesarias.

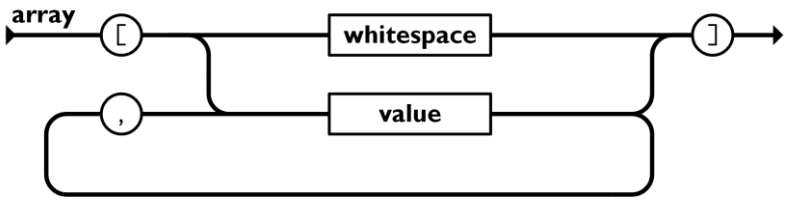


Diagrama sintático extraído de https://www.json.org

Entidades. Son a su vez un conjunto de atributos, se delimitan con los símbolos "**{"** y **"}"**, separando cada atributo con ",". Puede contener multitud de atributos.

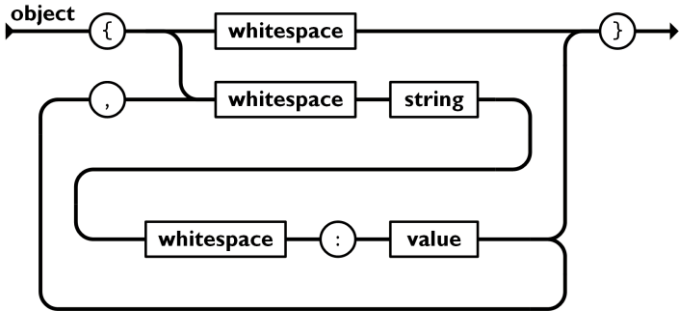


Diagrama sintático extraído de https://www.json.org

Atributos. Poseen un nombre y un valor, estos valores pueden ser de tipo primitivo o simples como texto o valores numéricos o más complejos como conjuntos de entidades y entidades, de forma que se puedan tener documentos con estructura en forma de árbol, al igual que en XML.

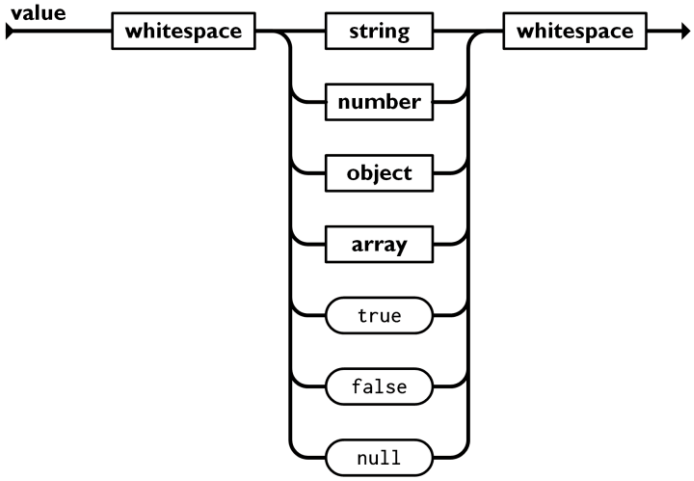


Diagrama sintático extraído de https://www.json.org

- **Los conjuntos de entidades son arrays y las entidades son objetos, pero no se han nombrado así para evitar confusiones con el módulo de programación.**

Un ejemplo de JSON:

```
{
  "juego": {
            "jugador": {
            "nombre": "Mario",
            "imagen": "mario.jpg",
            "vidas": 3,
            "poderes": [
                {
                    "nombre": "saltar"
                },
                {
                    "nombre": "disparar",
                    "balas": 3
                }
            ]
        },
        "enemigos": {
            "goombas": [
                {
                    "posicion": {
                        "x": 4,
                        "y": 10,
                        "inteligencia": 2
                    }
                },
                {
                    "posicion": {
                        "x": 4,
                        "y": 10,
                        "inteligencia": 2
                    }
                }
            ],
            "koopas": []
```

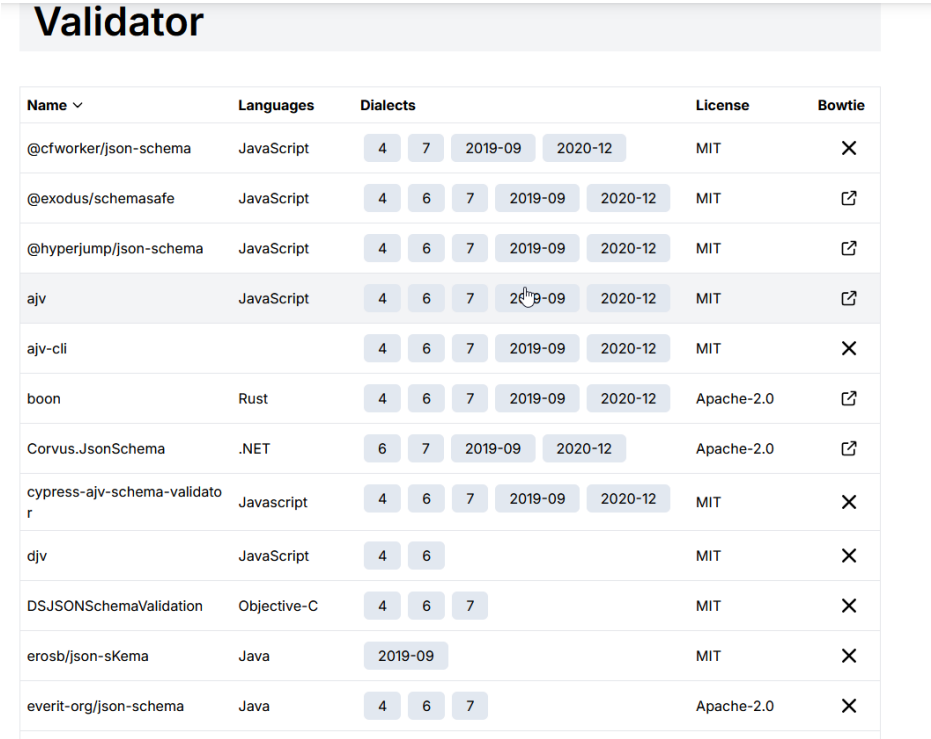
```
            }
        }
}
```

Existen múltiples librerías (incluso algunos lenguajes lo incluyen en su distribución estándar) y páginas en las que se puede validar un JSON, algunas de las páginas:

- https://jsonlint.com/
- https://jsonformatter.curiousconcept.com/

Inicialmente no existía una necesidad de implementar funcionalidad en JSON que si posee XML como la validación, la transformación o la búsqueda(la búsqueda es sencilla en JSON), pero han aparecido soluciones que las añaden.

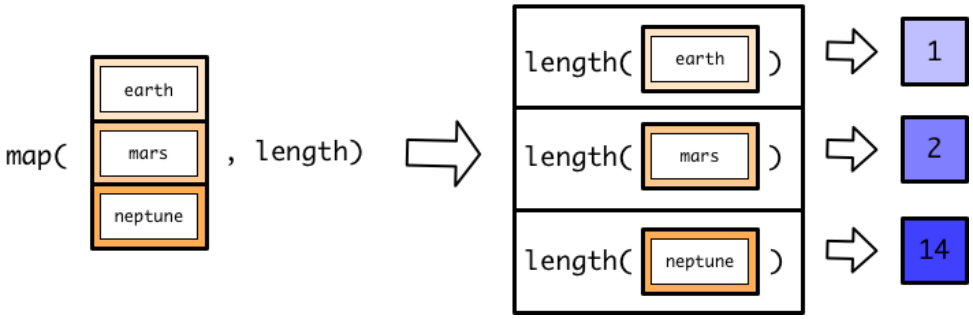
- **Validación.** Se denomina JSON Schema, existen librerías para prácticamente todos los lenguajes significativos de programación, se puede encontrar en https://json-schema.org/.



Validator

Name	Languages	Dialects	License	Bowtie
@cfworker/json-schema	JavaScript	4 7 2019-09 2020-12	MIT	✕
@exodus/schemasafe	JavaScript	4 6 7 2019-09 2020-12	MIT	
@hyperjump/json-schema	JavaScript	4 6 7 2019-09 2020-12	MIT	
ajv	JavaScript	4 6 7 2019-09 2020-12	MIT	
ajv-cli		4 6 7 2019-09 2020-12	MIT	✕
boon	Rust	4 6 7 2019-09 2020-12	Apache-2.0	
Corvus.JsonSchema	.NET	6 7 2019-09 2020-12	Apache-2.0	
cypress-ajv-schema-validator	Javascript	4 6 7 2019-09 2020-12	MIT	✕
djv	JavaScript	4 6	MIT	✕
DSJSONSchemaValidation	Objective-C	4 6 7	MIT	✕
erosb/json-sKema	Java	2019-09	MIT	✕
everit-org/json-schema	Java	4 6 7	Apache-2.0	✕

- **Transformación.** No existe una estandarización como en el caso de la validación, pero si existen herramientas no estándar para realizar la tarea como: JMESPath, JSONata o Jq entre otros. La transformación no suele ser una tarea común en JSON, y se puede solucionar en muchas ocasiones usando la programación funcional.



Ejemplo transformación usando programación funcional. Fuente https://dcl-prog.stanford.edu.

- Búsqueda. Se puede realizar usando el propio lenguaje de programación, aunque el estándar XQuery (lenguaje de consultas para XML), en su versión 3.1 permite procesar JSON de forma nativa.