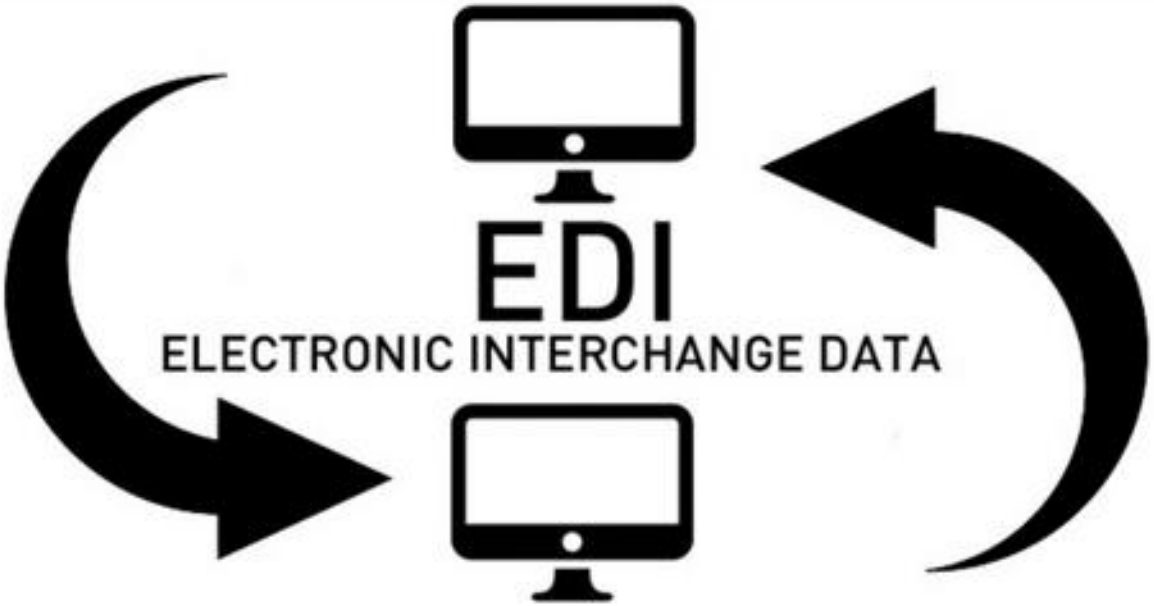
FORMATOS DE INTERCAMBIO DE DATOS



El intercambio electrónico de datos (en inglés electronic data interchange o **EDI**) es la transmisión estructurada de datos entre organizaciones por medios electrónicos.

Se usa para transferir documentos electrónicos o datos de negocios de un sistema computacional a otro.

Este intercambio puede realizarse en distintos formatos.

FORMATOS DE INTERCAMBIO DE DATOS

EDI es un formato estándar para intercambiar información entre dos organizaciones de forma electrónica en lugar de utilizar documentos en papel

```
UNB+UNOA+777+222+2702   :17+
99999'UNH+99999+ORDERS:1'BG
M+105+0001+270296'NAD+ST+++
mayor5'NAD+SE+++cerrada2'CUX
+ESP:OC'UNS+D'LIN+1++333:EN+
555+44444'DTM+002+280296'UNS
+S'UNT+10+99999'UNZ+1+99999'
```

Todas las transacciones de EDI se definen mediante estándares de mensajes de EDI.

En general, hay dos tipos básicos de transmisión de EDI:

- Conexiones de punto a punto o directas: dos computadoras o sistemas se conectan sin intermediarios a través de Internet, generalmente con protocolos seguros.
- Red de valor añadido (VAN): una red de terceros gestiona la transmisión de datos, generalmente con un paradigma de buzón de correo.

En transacciones de **EDI**, la información se traslada directamente desde una aplicación informática en una organización a una aplicación de sistema en otra. Los **estándares de EDI** definen la ubicación y el orden de la información en un formato de documento.

FORMATOS DE INTERCAMBIO DE DATOS

Estandares EDI: (Documentos EDI estándar)

Los estándares definen las reglas a tener en cuenta para la emisión e interpretación de una transacción concreta.

Diferente tipos de estándares EDI:

Los desarrollados específicamente para industrias concretas.
Los extendidos en más sectores o regiones.

Los más utilizados son:

Estándar EDI - UN/EDIFACT

United Nations/Electronic Data Interchange for Administration, Commerce and Transport, desarrollado por Naciones Unidas durante la década de los 80 gracias al trabajo del grupo Working Party 4.

Se trata de un lenguaje EDI multi país y multi industria, muy usado en Europa especialmente en el sector Retail, primera industria en adoptarlo, si bien con el paso de los años, su uso se ha generalizado a otros ámbitos como la sanidad, logística y transporte o construcción.

FORMATOS DE INTERCAMBIO DE DATOS

Los más utilizados son:

Estándar EDI - ANSI ASC X12

American National Standards Institute (ANSI) creó a finales de los 70 el Accredited Standards Committee (ASC)
Un estándar de mensajería uniforme para el intercambio de documentos electrónicos.
ANSI X12 inicialmente como un estándar a emplear por empresas norteamericanas de diferentes sectores.
Más información sobre ANSI X12

Estándar EDI - ODETTE

Organization for Data Exchange by Tele Transmission in Europe (ODETTE)
Representa los intereses de la industria automovilística en Europa
Aplica un protocolo propio de comunicación: OFTP (Odette File Transfer Protocol)

FORMATOS DE INTERCAMBIO DE DATOS

Los más utilizados son:

Estándar EDI - UBL (Universal Business Language)

Basado en XML, diseñado inicialmente para el sector financiero.
La Unión Europea, en el marco de su estrategia 2020, ha designado el UBL como el estándar de referencia para la contratación pública como base para definir los mensajes que deben intercambiar con sus proveedores.

Por supuesto que existen otros Estándares Internacionales y Estándares que Gestionan los Gobiernos de cada Pais.

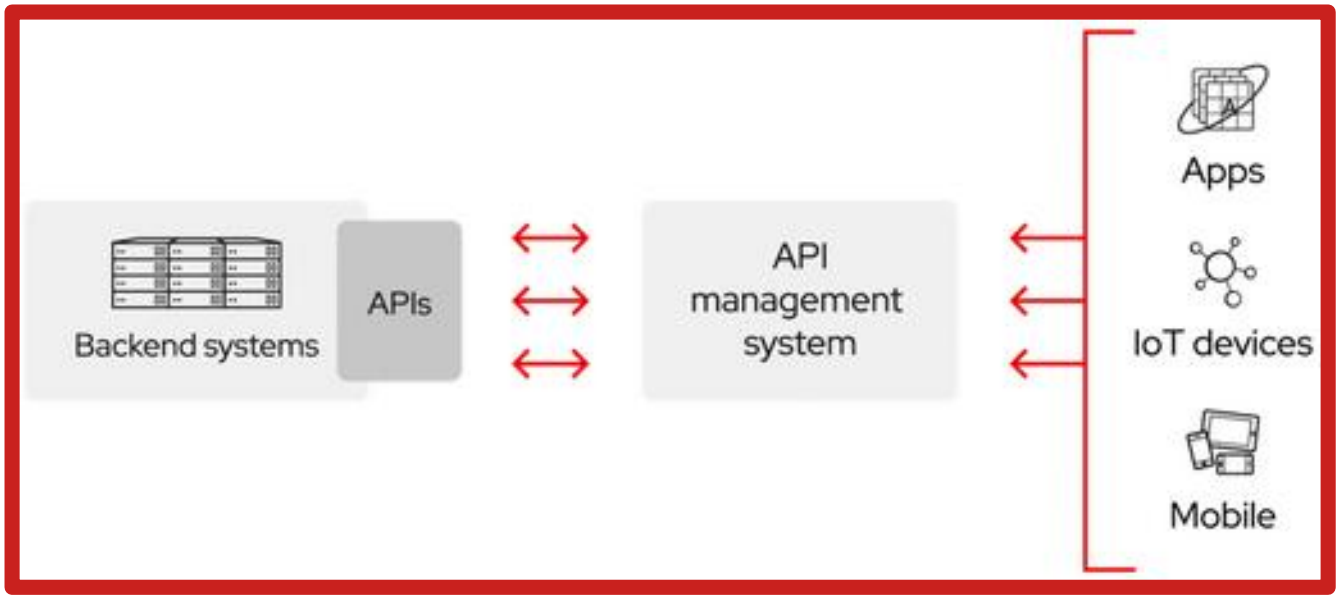
También Existen Estándares dentro de una misma Organización.

Ejemplo: Relación de los Sistemas del SIU con los Sistemas de la Universidades.

FORMATOS DE INTERCAMBIO DE DATOS

API

API son las siglas de **Aplication Programming Interface**, en español Interfaz de Programación de Aplicaciones.
Se trata de un conjunto de reglas que va a definir cómo se comunican dos aplicaciones entre sí.



FORMATOS DE INTERCAMBIO DE DATOS

API

API son las siglas de **Aplication Programming Interface**, en español Interfaz de Programación de Aplicaciones.
Se trata de un conjunto de reglas que va a definir cómo se comunican dos aplicaciones entre sí.

Los tres tipos básicos de APIs

APIs toma tres formas básicas: local, como web y como programa. Aquí hay un vistazo a cada tipo

Local APIs	Web APIs	Programa APIs
La API original, creada para proporcionar servicios de sistema operativo o middleware a los programas de aplicación	Diseñado para representar recursos ampliamente utilizados como páginas HTML y se accede mediante un protocolo HTML simple. A menudo llamadas APIs REST vs APIs RESTFUL	Basado en la tecnología RCP que hace que un componente de programa remoto parezca ser local al resto del software

FORMATOS DE INTERCAMBIO DE DATOS

Actualmente las API web normalmente usan **HTTP/s** para solicitar mensajes y proporcionar una definición de la estructura de los mensajes de respuesta.
Generalmente, las respuestas toman la forma de un archivo **XML** o **JSON** Formatos preferidos por lo fácil de manejar para otras aplicaciones.

Se basan en Servicios contemplados en **SOA**.

La arquitectura orientada a los servicios (SOA) es un tipo de diseño de software que permite reutilizar sus elementos gracias a las interfaces de servicios que se comunican a través de una red con un lenguaje común.

Protocolos de API remotas (SOAP)

Protocolo de Acceso a Objetos Simples (SOAP):

Usan **XML** para el formato de sus mensajes.

Reciben solicitudes a través de **HTTP** o **SMTP**.

SOAP, facilita que la comunicación entre aplicaciones que funcionan en entornos distintos o están escritas en diferentes lenguajes compartan información.

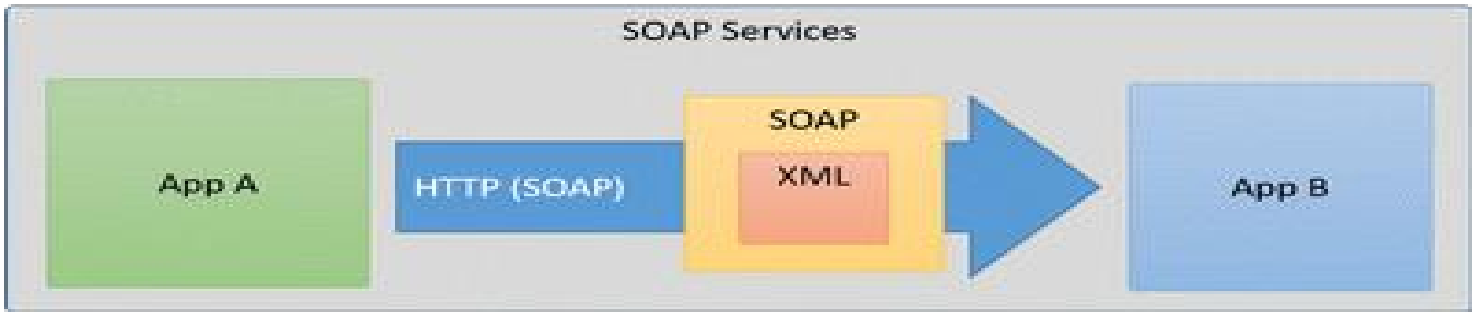
Transferencia de Estado Representacional (REST):

Las API web que funcionan con las limitaciones de arquitectura **REST** se llaman **API de RESTful.**

Usan **JSON** para el formato de sus mensajes.

No hay ningún estándar oficial para las API web de RESTful.

FORMATOS DE INTERCAMBIO DE DATOS



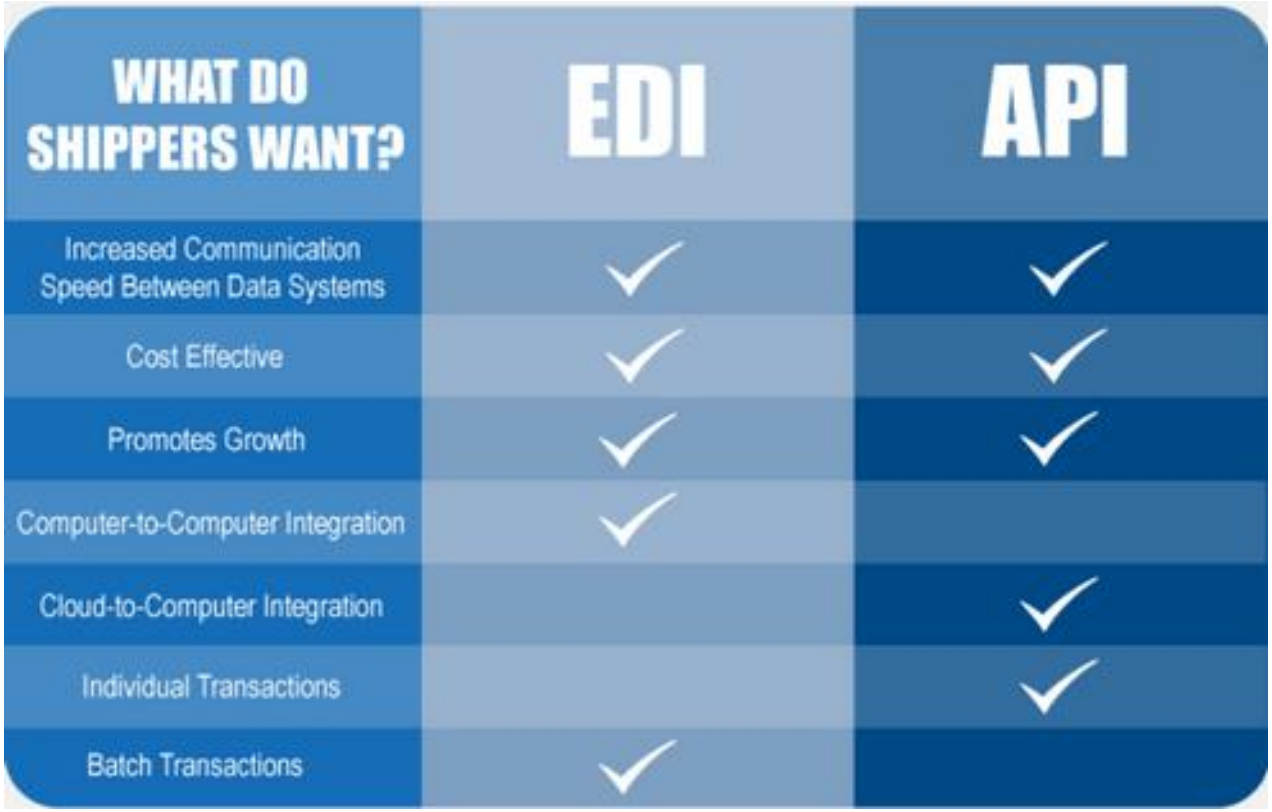
Permite: XML **HTTP, FTP, POP3, TCP, etc**

Permite: XML, JSON, Binarios (imágenes, documentos), Text, etc.

FORMATOS DE INTERCAMBIO DE DATOS

EDI frente a **API**

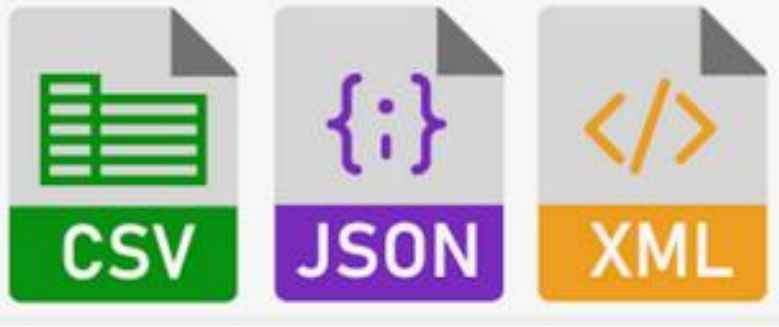
Mientras que los EDIs tienen grandes medidas de fiabilidad y seguridad, las **API** son practicamente ilimitadas cuando se trata de los tipos de sistemas con los que se puede interconectar.



FORMATOS DE INTERCAMBIO DE DATOS

Volvamos a los Datos...

1) **XML**
2) **JSON**
3) YAML
4) CSV
5) Etc



Estos formatos estan pensados para intercambiar datos entre sistemas de dististintos lenguajes. Otra forma de compartir datos, pero no entre sistemas, son con documentos.

CSV

Las siglas CSV vienen del inglés "Comma Separated Values" y significan **valores separados por comas**.
Un archivo CSV es cualquier archivo de texto en el cual los caracteres están separados por comas, haciendo una especie de **tabla en filas y columnas**. Donde las columnas quedan definidas por cada punto y coma (;).

Nota: Esto es adaptable a sus necesidades. (puedo seleccionar otro carácter especial de limitación.

Los archivos CSV sirven para manejar una gran cantidad de datos en formato tabla.

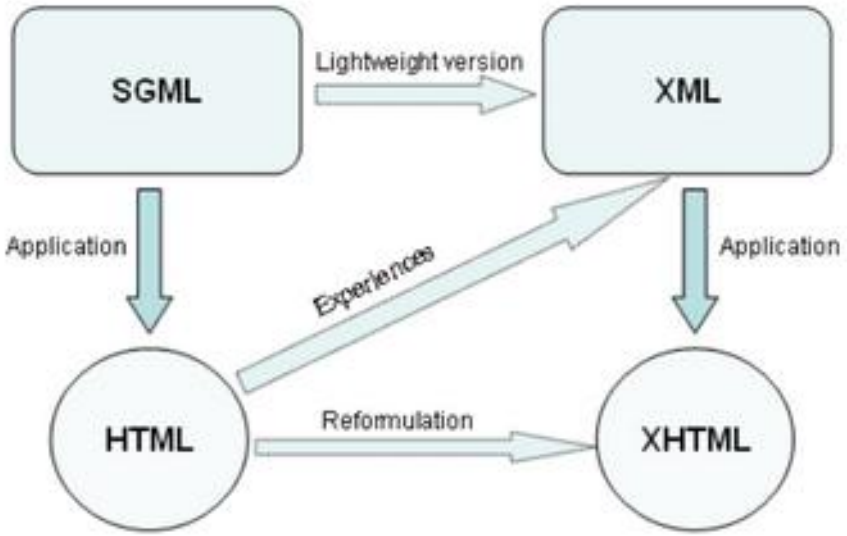
Limitaciones:

los archivos CSV tan solo admiten datos, es decir, no podremos dar formato a los mismos.
La capa de presentación la debe resolver el programa receptor.

FORMATOS DE INTERCAMBIO DE DATOS

SGML:

SGML es un acrónimo de Standard Generalized Markup Language o Lenguaje de Señalización General Normalizado.
Esta es una norma ISO derivada de una anterior (**GML de IBM**).
SGML permite que la estructura de un documento pueda ser definida en base a la relación lógica de sus partes.



HTML *(Hyper Text Markup Language)*
Lenguaje de marcas de texto.
utilizado normalmente en la **www.**

XHTML Junta la capacidad de formato de HTML Consolida con la formalidad del XML (y sus reglas).
Permite ser Validado por herramientas Estandar.

XML


```
1 <persona>
2     <nombres>Elsa</nombres>
3     <apellidos>Zambrano</apellidos>
4     <fecha-de-nacimiento>
5         <día>18</día>
6         <mes>6</mes>
7         <año>1996</año>
8     </fecha-de-nacimiento>
9     <ciudad>Pamplona</ciudad>
10</persona>
```

SGML es un acrónimo de Standard Generalized Markup Language o Lenguaje de Señalización General Normalizado. Esta es una norma ISO derivada de una anterior (GML de IBM). **SGML** permite que la estructura de un documento pueda ser definida en base a la relación lógica de sus partes.

Que es el XML?

XML es un subconjunto de
SGML (Estándar Generalised Mark-up Language), simplificado y adaptado a Internet.

XML es un meta-lenguaje que nos permite definir lenguajes de marcado adecuados a usos determinados.

XML

¿Para que sirve XML?

Representar información estructurada en la web (todos documentos), de modo que esta información pueda ser almacenada, transmitida, procesada, visualizada e impresa, por muy diversos tipos de aplicaciones y dispositivos.

XML No es una versión mejorada de **HTML**
HTML es una aplicación de **SGML** por lo tanto de
XML No es un lenguaje para hacer paginas WEB

Problema del HTML

- Define mas la presentación que su contenido
- No es fácilmente procesables por las maquinas
- Su estructura es caótica
- Su interpretación es ambigua dependiendo S.W utilizado
- Solo vale para paginas WEB

Ventajas del XML

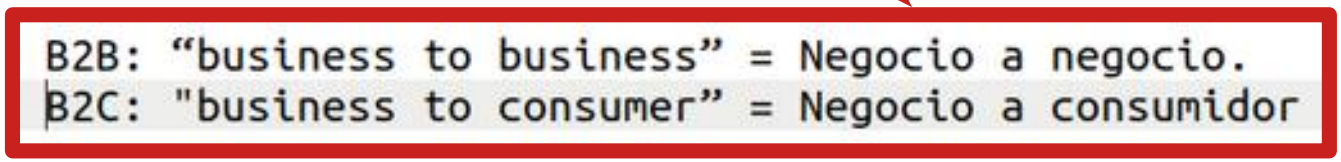
- Fácilmente procesable
- Separa radicalmente el contenido y el formato de presentación
- Diseñado para cualquier lenguaje y alfabeto. (encoding)

Características

- XML es un subconjunto de SGML que incorpora sus tres características más importantes:
  - Extensibilidad
  - Estructura
  - Validación
- Basado en texto.
- Orientado a los contenidos no presentación.
- Las etiquetas se definen para crear los documentos, no tienen un significado preestablecido.
- No es sustituto de HTML.
- No existe un visor genérico de XML.

Aplicaciones de XML

- Publicar e intercambiar contenidos de bases de datos.
- Formatos de mensaje para comunicación entre aplicaciones (B2B)
- Descripción de metacontenidos.



Documento XML

- Conjunto de datos con sus respectivas etiquetas de marcado XML.
- Se almacena como texto en archivo con extensión .xml.
- Un documento XML puede incluir cualquier flujo de datos basado en texto:
  - un articulo de una revista
  - un resumen de cotizaciones de bolsa
  - un conjunto de registros de una base de datos,
  - etc..

Estructura de un documento XML

- Un documento XML está formado por **datos de caracteres** y **marcado**, el marcado lo forman las etiquetas.
- Dentro del documento existen dos partes: el **Prologo** y el **Cuerpo**

Prologo:

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="no" ?>
<!DOCTYPE persona SYSTEM "persona.dtd">
```

Cuerpo:

```
<persona>
    <nombre> Franco  </nombre>
    <apellido> Kühn </apellido>
</persona>
```

Estructura Básica

La estructura esta compuesta por:

- Elementos
- Atributos
- Contenido del elemento

Componentes de un documento XML

- **En un documento XML existen los siguientes componentes:**
  - **Elementos**: Pieza lógica del marcado, se representa con una cadena de texto(dato) encerrada entre etiquetas. Pueden existir elementos vacíos (
). Los elementos pueden contener atributos.
  - **Instrucciones**: Ordenes especiales para ser utilizadas por la aplicación que procesa

<?xml-stylesheet type="text/css" href="estilo.css">

  - **Las instrucciones XML**. Comienzan por <? Y terminan por ?>.
  - **Comentarios**: Información que no forma parte del documento. Comienzan por: <!-- y terminan por -->.
  - **Declaraciones de tipo**: Especifican información acerca del documento:

<!DOCTYPE persona SYSTEM "persona.dtd">

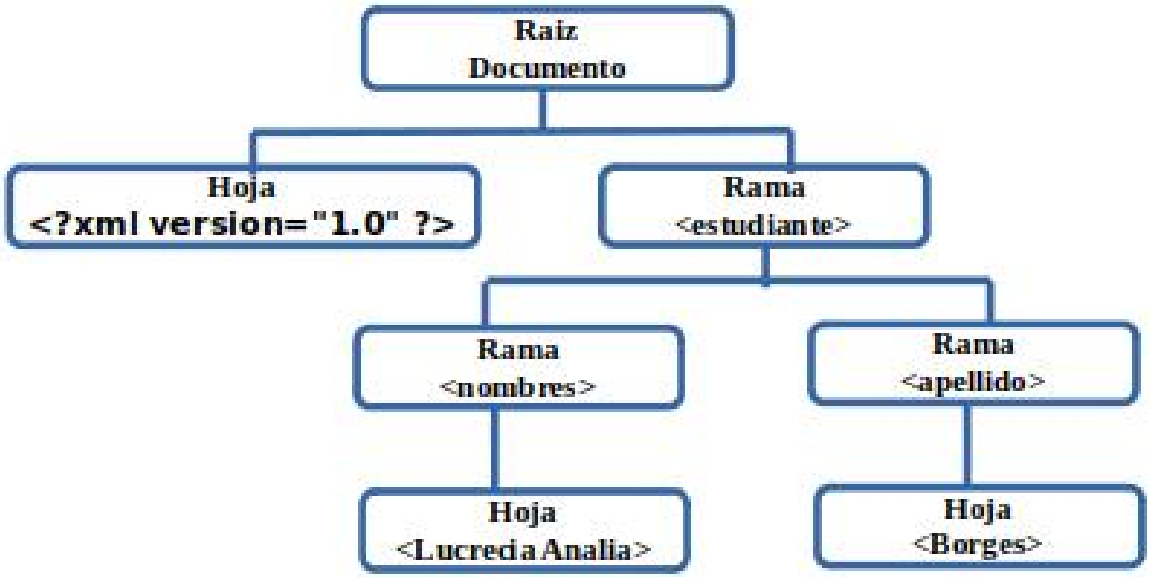
  - **Secciones CDATA**: Se trata de un conjunto de caracteres que no deben ser interpretados por el procesador:

<![CDATA[Aquí se puede meter cualquier carácter, como <, &, >, ... Sin que sean interpretados como marcación]]>

Árbol de Análisis

- Estructura que muestra los objetos que forman el documento y las relaciones entre ellos.
- Los componentes de un documento se les llama **objetos** (elementos, comentarios y cadenas de texto). El propio documento es un objeto.
- A cada objeto del árbol se le denomina **nodo**.
- El nodo principal que contiene a los demás se le llama nodo **raiz**.
- Cuando un nodo contiene a otro se le denomina **rama**.
- Los nodos finales, que no contienen otros nodos, se llaman **hojas**.

```
<?xml version="1.0" ?>
<estudiantes><persona>
<nombres> Lucrecia Analia </nombre>
<apellido > Borges </apellido>
```



Aclaración

- A la hora de diseñar se nos plantean dudas entre que escoger atributo o elemento. Hay que tener en cuenta lo siguiente:
  - **los atributos no pueden contener subelementos ni subatributos.**
  - No se organizan en ninguna jerarquía por lo que la representación es mucho más reducida que los elementos
  - La utilización de los atributos será una mera modificación de los elementos a que se aplican la información que deben de contener
    debe de ser de poca entidad sencilla y sin estructura.
- Aun así muchas veces llegamos a la misma conclusión utilizando atributos y elemento

Definición y Validación de XML (Reglas de Construcción)

- Un documento XML puede:
  - **Estar bien formado**: Cumple con la **sintaxis de XML.**
- O puede:
  - **Ser válido**: Además de estar bien formado, el documento cumple unas determinadas reglas y normas.
- Un analizador que comprueba la validez de un documento se dice que es un **analizador de validación**.
- Para establecer las reglas de construcción de documentos XML se utilizan **las DTD's** y los **Esquemas XML**.

Para definir y validar XML se puede utilizar las:

DTD

y/o

Esquemas XML

Definición de Tipo de Documento (DTD)

Define los elementos, atributos, entidades y notaciones que pueden utilizarse para construir un tipo de documentos, así como las reglas para su utilización.

- Mediante ellas se comprueba la validez de un documento.
- Tienen su origen en SGML.
- Posee una sintaxis especializada.

Ejemplo:

```
<!ELEMENT persona (nombre, apellidos, teléfono*)>
<!ELEMENT nombre (#PCDATA)>
<!ELEMENT apellidos (#PCDATA)>
<!ELEMENT teléfono (#PCDATA)>
```

- La DTD establece la gramática de un vocabulario XML, el cual, determina la estructura de los documentos XML.
- Para que un documento se adhiera a una DTD, deberá incluir en su cabecera una declaración de tipo de documento:

```
<!DOCTYPE ElementRaiz SYSTEM dtd>
```

  - ElementRaiz identifica el elemento de documento.
  - Dtd representa la definición de la DTD.
- La declaración de tipo de documento, debe aparecer después de la declaración XML y antes del primer elemento de un documento.

La Declaración de Tipo de Documento (DTD-Document Type Data):

Al definir el lenguaje XML ya nos referimos a la Definición del Tipo de Documento (Document Type Definition DTD) que, en resumen, cumple las siguientes funciones:

- Una DTD especifica la clase de documento
  - Describe un formato de datos
  - Usa un formato común de datos entre aplicaciones
  - Verifica los datos al intercambiarlos
  - Verifica un mismo conjunto de datos

- Una DTD describe
  - Elementos: cuáles son las etiquetas permitidas y cuál es el contenido de cada etiqueta
  - Estructura: en qué orden van las etiquetas en el documento
  - Anidamiento: qué etiquetas van dentro de cuáles

Declaraciones de elemento

- Para declarar los elementos, que pueden ser utilizados por los documentos que se ajusten a esa **DTD**, se utiliza la expresión:

<!ELEMENT NombreElemento Contenido>

- *Contenido*, representa el contenido del elemento y puede ser:
  - **Una lista de elementos secundarios.**
    Cuando un elemento está formado por otros elementos, estos se declaran entre paréntesis separados por comas o por | si se trata de elementos alternativos:

<!ELEMENT planeta (nombre+, tipo, (orbita|satélite+), composición*)>

    - Cada subelemento puede llevar, además, los siguientes símbolos:
      – **Sin símbolo**: El elemento secundario debe aparecer una sola vez.
      – **Interrogación**: Puede aparecer una o ninguna vez.
      – **Asterisco (*)**: Puede aparecer cualquier número de veces.
      – **Signo más (+)**: Debe aparecer por lo menos una vez.
  - **Datos de carácter**. Cuando un elemento solo va a contener datos de carácter analizados sintácticamente, se utiliza la declaración (#PCDATA):

<!ELEMENT nombre (#PCDATA)>

    - Es posible declarar elementos mixtos, es decir, que contengan datos de **carácter y elementos secundarios:**

<!ELEMENT capitulo (#PCDATA|subcapitulo)*>

  - **Elemento vacío**. Los elementos sin contenido se declaran:

<!ELEMENT saltolinea EMPTY>

Declaraciones de atributos

- Los atributos se utilizan para especificar información adicional del elemento.

```
<libro>
  <capitulo numero="1">
      <subcapitulo>
            :
```

- Se declaran utilizando la siguiente sintaxis:

<!ATTLIST Elemento *NomAtr Tipo Valpred*>

- ***NomAtr***. Es el nombre que se le da al atributo.
- ***Tipo***. Es el tipo de atributo. Puede tomar uno de los siguientes valores:
  - **CDATA**: Datos de caracteres no analizados sintácticamente.
  - **Lista de valores**: Entre paréntesis y separados por |. Ejemplo: (Este | Oeste | Sur | Norte)
  - **ENTITY**: Entidad binaria externa.
  - **NOTATION**: Notación declarada en la DTD.
  - **ID**: Identificador único.
- ***Valorpred***. Representa el valor predeterminado del atributo. Puede ser una de las siguientes posibilidades:
  - #REQUIRED: El atributo es obligatorio.
  - #IMPLIED: El atributo es opcional.
  - #FIXED valor: El atributo tiene ese valor fijo.
  - Un valor.

Declaraciones de atributos

Un elemento puede tener más de un atributo y pueden ser declarados todos en la misma línea:

```
<!ATTLIST curso
director CDATA #REQUIRED
horario (mañana | tarde | noche)
#IMPLIED
instalaciones (Exes | externas)
“Exes”>
```

```
<!ELEMENT addressbook (contact)+>
<!ELEMENT contact (name, address+, city, state, zip, phone, email, web, company)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT address (#PCDATA)>
<!ELEMENT city (#PCDATA)>
<!ELEMENT state (#PCDATA)>
<!ELEMENT zip (#PCDATA)>
<!ELEMENT phone (voice, fax?)>
<!ELEMENT voice (#PCDATA)>
<!ELEMENT fax (#PCDATA)>
<!ELEMENT email (#PCDATA)>
<!ELEMENT web (#PCDATA)>
<!ELEMENT company (#PCDATA)>
<!ELEMENT peliculas (pelicula)+>
<!ELEMENT pelicula(titulo, writer+, productor+, director+, actor*, comentarios?)>
<!ATTLIST pelicula
tipo (drama | comedy | adventure | sci-fi | mistery | horror | romance | documentary)
"drama"
clasificacion (G | PG | PG-13 | R | X) "PG"
review (1 | 2 | 3 | 4 | 5) "3"
año CDATA #IMPLIED>
<!ELEMENT titulo(#PCDATA)>
<!ELEMENT writer (#PCDATA)>
<!ELEMENT productor(#PCDATA)>
<!ELEMENT director (#PCDATA)>
<!ELEMENT actor (#PCDATA)>
```

Ejemplo de DTD

Recordemos que una DTD se puede guardar en un archivo de texto, como por ejemplo, en el archivo "list.dtd".:

```
<!ELEMENT List (Item)+>
<!ELEMENT Item (#PCDATA)>
<!ATTLIST Item
id CDATA IMPLIED
color CDATA IMPLIED>
<!ELEMENT Separator EMPTY>
```

Ejemplo de uso de DTD

Veamos ahora el documento .xml que hace referencia a esa DTD:

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE list SYSTEM "list.dtd">
<List>
<Item>París</Item>
<Item>Madrid</Item>
<Separator />
<Item color="rojo">Londres</Item>
</List>
```

Esquemas XML

- Misma finalidad que las **DTD's**.
  Describen los elementos y atributos que se pueden utilizar para construir documentos XML y las reglas de utilización.
- Permiten asociar tipos de datos con los elementos.
- Se crean utilizando la sintaxis XML.

Ejemplo:

```
<Schema xmlns="urn:schemas-microsoft-com:xml-data"
xmlns:dt="urn:schemas-microsoft-com:datatypes">
    <AttributeType name='id' dt:type='string' required='yes'/>
    <ElementType name='nombre' content='textOnly'/>
    <ElementType name='persona' content='mixed'>
        <attrubyte type='id'/>
        <element type='nombre'/>
    </ElementType>
    <ElementType name='documento' content='eltOnly'>
        <element type='persona'/>
    </ElementType>
</Schema>
```

Fundamentos

- Surgen para dar respuesta a las limitaciones de las DTD's.
- Se basa en el vocabulario XML-Data que se utiliza para describir la estructura de los documentos.
- A la tecnología para la creación de esquemas se le conoce como XML Schema.
- IE 5.0 no realiza validación del documento a través del esquema.

Generalidades

- Los esquemas se definen en documentos XML.
- Se hace referencia a ellos a través de los espacios de nombres:

```
<peliculas xmlns="x-schema:PeliculasEsquema.xml">
```

Espacios de nombres

- Un espacio de nombres es la dirección (URL o URN) que contiene la DTD o esquema, utilizado para validar los elementos del documento.
- Permiten evitar el conflicto de nombres dentro de un documento XML.

```
<peliculas
  xmlns:pelicula="http;//www.aplicaciones.com/peliculas"
  xmlns:musica="http://www.aplicaciones.com/musica"
  <!-- Contenido del documento-->
</peliculas>
```

Elementos de un Esquema

- **Schema**. Es el elemento raíz del documento, actúa como contenedor para el resto de los elementos. Contiene los atributos:
  - **name**: Nombre del esquema.
  - **xmlns**: Espacio de nombres del esquema. Hace referencia a la DTD donde se definen los elementos del esquema. Su valor se establece a:

  urn:schemas-microsoft-com:xml-data
- **ElementType**. Define los tipos de elementos que se utilizarán para la elaboración de documentos que sigan el esquema. Contiene los atributos:
  - **name**: Nombre del elemento.
  - **content**: Contenido del elemento:
    - eltOnly: Sólo puede contener elementos secundarios.
    - textOnly: Sólo texto.
    - mixed: Texto y elementos secundarios.
    - empty: Sin contenido.
  - **order**: Orden y frecuencia del grupo de elementos secundarios del elemento:
    - One. Sólo se permite una serie de elementos.
    - Seq. Los elementos deben producirse en la secuencia especificada.
    - Many. Los elementos pueden aparecer las veces que sea en cualquier orden.
- **dt:type**. Establece el tipo de contenido del elemento.

Elementos de un Esquema

- **Element**. Declara el modelo de contenido para un elemento. Dispone de los atributos:
  - **type**. Tipo de elemento. Es el nombre con el que ha sido declarado en ElementType.
  - **minOccurs**. Mínimo número de veces que el elemento puede aparecer. Su valor puede ser 0 o 1.
  - **maxOccurs**. Máximo número de veces que el elemento puede aparecer. Puede ser 1 o *.

```
<ElementType name="location" content="textOnly"/>
<ElementType name="comments" content="textOnly"/>
<ElementType name="session" content="eltOnly" order="seq">
 <element type="location" minOccurs="1" maxOccurs="1"/>
 <element type="comments" minOccurs="0" maxOccurs="1"/>
</ElementType>
```

- **AttributeType**. Se utiliza para definir los tipos de atributos que van a ser utilizados en los elementos. Los atributos son:
  - **name**: Nombre del atributo.
  - **dt:type**: Tipo de datos del atributo.
  - **dt:values**: Lista posible de valores de un atributo enumerado. Sólo se aplica cuando dt:type está establecido a enumeration.
  - **default**: Valor predeterminado.
  - **required**: Indica si el atributo es o no obligatorio. Su valor es **yes** o **no**.
- **attribute**. Se utiliza para declarar el modelo de contenido de un elemento. Sus atributos son:
  - **type**. Nombre del atributo
  - **default**. Valor predeterminado
  - **required**. Si es o no obligatorio

Tipos de datos de XML Schema

- Los tipos de datos de XML Schema están definidos en el espacio de nombres urn:schema-microsoft-com:datatypes.
- Entre los tipos de datos más importantes están:
  - **char**. Carácter.
  - **string**. Cadena de caracteres.
  - **boolean**. Booleano 0 o 1.
  - **int**. Número entero.
  - **float**. Número real.
  - **date**. Fecha.
  - **time**. Hora
  - **uri**. Identificador Uniforme de Recursos
  - **enumeration**. Tipo enumerado, sólo válido para atributos.
  - **ID**. Atributo de tipo identificador

Estándares relacionados con XML: xpath, xpointer, xlink, xslt, DOM.

- El XPath es un lenguaje de expresión que permite procesar valores confome a los modelos de datos XML.
- XPointer (the XML Pointer language) permite armar hipervinculos que apuntan a partes especificas de fragmentos de documentos XML.
- XLink (the XML Linking language) define el metodo para crear links dentro de los documetos XML.
- XSLT soportar las transformaciones XSL. El XSLT permite las tranformaciones de documetos XML a otros formatos como XHTML.
- El dom xml define el estandar para acceder y manipular los documentos XML. El DOM presneta un documento XML com una estructura de arbol.
  - El DOM es separdo por la W3C en tres diferentes partes
    - CORE DOM: define el modelo estandar para cualquier documento estrucutrado
    - XML DOM: el modelo estandar para documentos XML.
    - HTML DOM: el modelo estadar para documentos HTML

Algunos sitios para leer

- http://www.w3schools.com/xpath/
- http://es.wikipedia.org/wiki/XPath
- http://www.w3.org/TR/xpath20/
- http://www.w3schools.com/xml/xml_xpointer.asp
- http://www.w3schools.com/xml/xml_xlink.asp
- http://www.w3.org/TR/xlink11/
- http://www.w3schools.com/xsl/
- http://www.w3.org/TR/xslt20/
- http://www.w3schools.com/dom/

Ejemplo sencillo
Ejemplo de Planilla de calculo
Ejemplo de Procesador de texto
Ejemplo de Bases de Datos mysql

Herramientas open-source para procesar XML.

butterflyXML (http://sourceforge.net/projects/butterflyxml) y conglomerate (http://www.conglomerate.org/), ambas con soporte para Docbook. El problema de ambas herramientas es que las últimas versiones estables son algo antiguas.

También está Quanta+, que es una herramienta de desarrollo de páginas web para KDE y forma parte del paquete kdewebdev. Es ampliable mediante plugins.

Otra opción libre es el editor XMLcopyeditor disponible tanto para GNU/Linux como Windows.

Herramientas open-source para procesar XML: xerces, xalan, fop, etc

JSON

JSON (JavaScript Object Notation

Notación de Objetos de JavaScript) es un formato ligero de intercambio de datos.

Alternativa para el **XML**. Es mas simple ya que esta construido por dos estructuras:

- Una colección de pares de nombre/valor. En varios lenguajes esto es conocido como un objeto, registro, estructura, diccionario, tabla hash, lista de claves o un arreglo asociativo.

- Una lista ordenada de valores. En la mayoría de los lenguajes, esto se implementa como arreglos, vectores, listas o sequencias.

Es por esto que es mas simple el parser. De hecho en JavaScript se puede utilzar el eval().

Definido por:

- ECMA-404 The JSON Data Interchange Standard.
- RFC - 7159

JSON

La sintaxis de intercambio de datos JSON

Un texto JSON es una secuencia de tokens formada a partir de puntos de código Unicode que se ajusta al valor JSON gramática.

El conjunto de tokens incluye seis tokens estructurales, cadenas, números y tres tokens de nombre literal.

[ ] { } : ,

Valores JSON

Un valor JSON puede ser un:

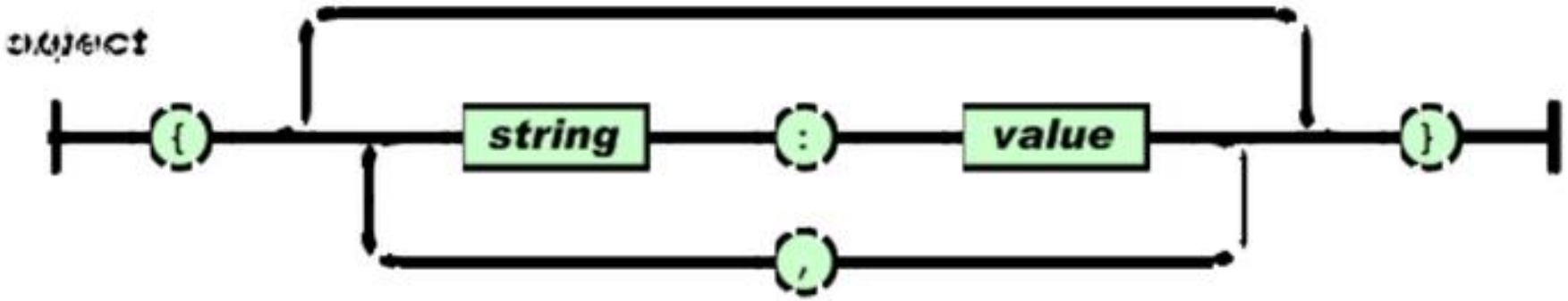
- objeto
- array
- número
- string
- true
- false
- null

Objeto:

Es un conjuto desordenados de pares nombre/valor. Comienza con llave de apertura ({) y termina con llave de cierre(}) Cada nombre es seguido por dos puntos (:) y los pares son separados por comas (,).

Ejemplo:

```
{
    "menu": {  "id": "file",  "value": "File", },
    "activado": 1
}
```

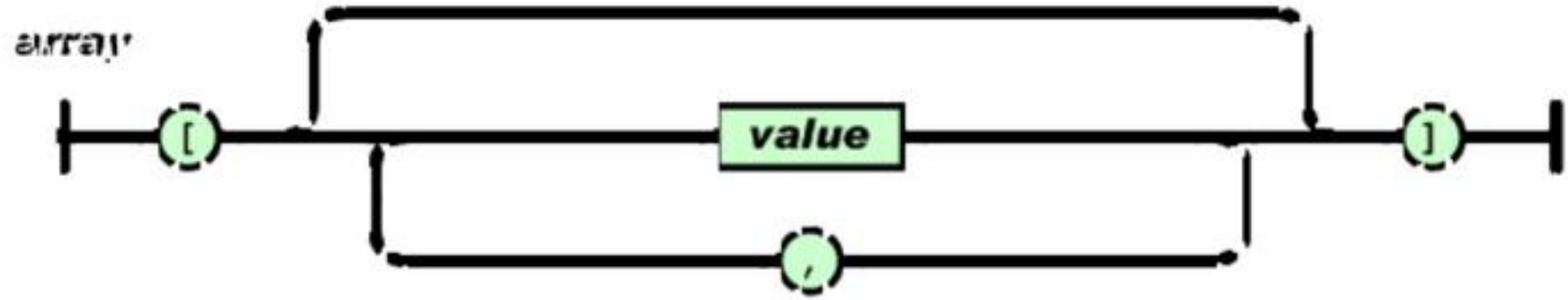


Arreglo:

Es una colección de valores. El arreglo empieza con corchete ([ ) y termina con el corchete de cierre ( ]). Los valores se separan por ,

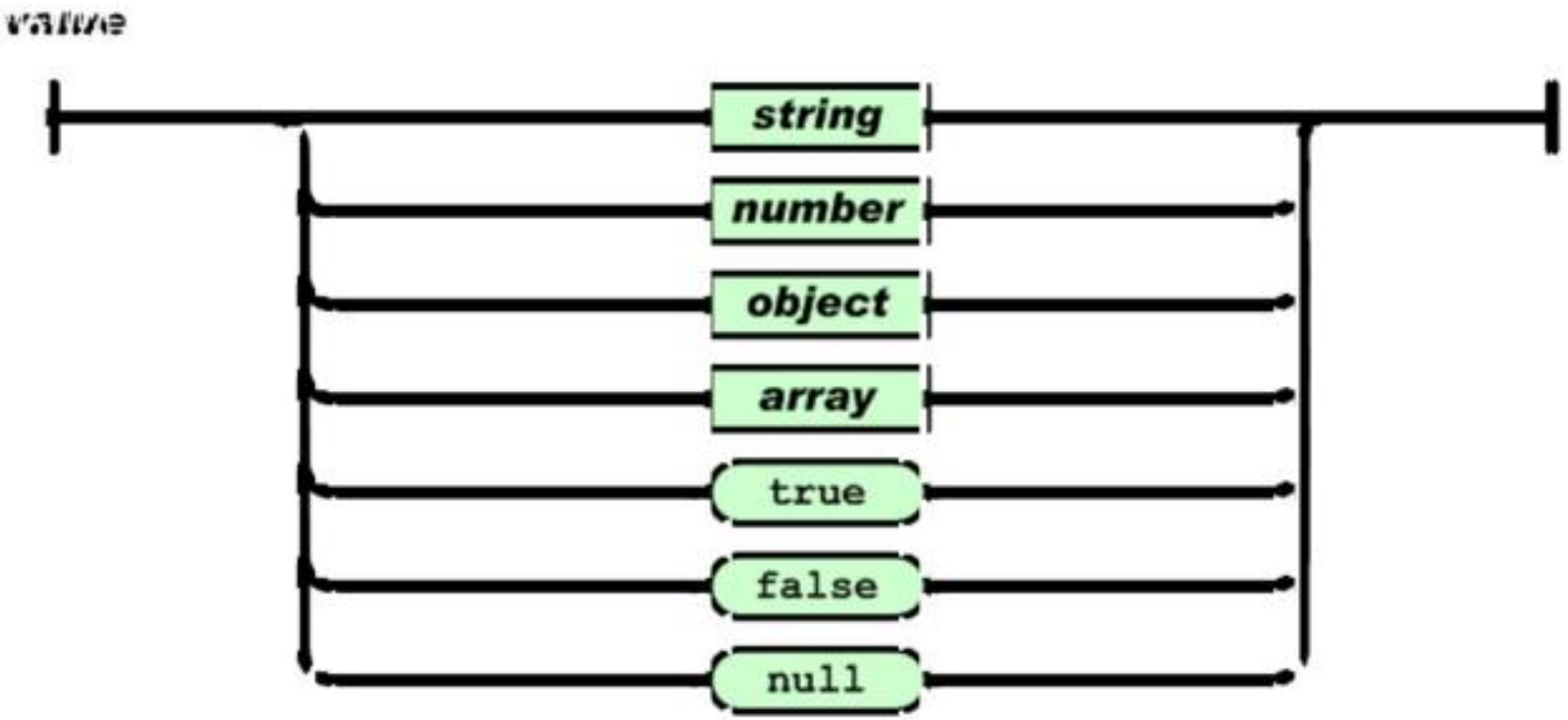
Ejemplo:

```
[
      “Valor1”,
       “Valor2”
]
```



Valor:

Un valor puede ser una cadena de caracteres con comillas dobles, o un número, o true o false o null, o un objeto o un arreglo. Estas estructuras pueden anidarse.



JSON

```
{
  "nombre":"Franco Kün",
  "profesion":"Ing. Informático",
  "edad":35,
  "lenguajes":["PHP","Javascript","C++"],
  "disponibilidadParaViajar":true,
  "rangoProfesional": {
      "aniosDeExperiencia": 5,
      "nivel": "Senior"
  }
}
```

https://www.w3schools.com/js/js_json_intro.asp

Algo mas sobre JS JSON

JSON Example

```
{"employees":[
  { "firstName":"John", "lastName":"Doe" },
  { "firstName":"Anna", "lastName":"Smith" },
  { "firstName":"Peter", "lastName":"Jones" }
]}
```

XML Example

```
<employees>
  <employee>
    <firstName>John</firstName> <lastName>Doe</lastName>
  </employee>
  <employee>
    <firstName>Anna</firstName> <lastName>Smith</lastName>
  </employee>
  <employee>
    <firstName>Peter</firstName> <lastName>Jones</lastName>
  </employee>
</employees>
```

JSON

JSON.Parse(cadena)

Esta función permite convertir una cadena en formato JSON en una variable Javascript que contiene la representación del valor de esa cadena.

```
var objeto = JSON.parse('{"nombre": "juan",
"edad": 33, "casado": false}');
```

JSON.stringify(objeto)

Esta función Javascript convierte un objeto, un array u otro valor en su representación como cadena en formato JSON.

```
var coche = {
  modelo: "Ford Focus",
  anioFabricacion: "2020",
  motorizacion: 'Gasolina'
}
var cadena = JSON.stringify(coche);
```

https://desarrolloweb.com/home/json#track68

Bibliografia:

- **json.org**
- **Json parser for C**
- **Json parser for C++**
- **JSON y BSON en MongoDB**
- **JSON en mysql**

YAML

YAML es un acrónimo recursivo que significa YAML Ain't Markup Language (en castellano, '**YAML no es un lenguaje de marcado**').

```
=== Listas ===

<syntaxhighlight lang="yaml">
 --- # Películas favoritas, formato de bloque
 - BotijoAzul
 - BotijoVerde
 - Viridiana
 - Psicosis
 --- # Lista de la compra, formato en línea
 [leche, pan, huevos]
 [chorizo, morcilla, botijo, pollo]
</syntaxhighlight>
```

```
=== Vectores asociativos ===
<syntaxhighlight lang="yaml">
 --- # Bloque
 nombre: Pepe López
 edad: 33
 --- # En  línea
 {nombre: Pepe López, edad: 33}
</syntaxhighlight>
```

Programas compatibles con el archivo YAML

1. Adobe Dreamweaver.
2. gVim.
3. NotePad++ text editor.
4. Python.
5. Windows Notepad.