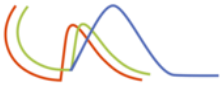


Contenido

FRONT-END

El desarrollo de aplicaciones web front-end hace referencia a las tecnologías que interactúan con el cliente, englobadas en el término "**del lado del cliente**". Su misión es mostrar al usuario la información, recibir sus peticiones y trasladarlas (de alguna manera) al servidor web para ser procesadas y dar una respuesta.

Las tecnologías web involucradas en el desarrollo del lado del cliente son, principalmente, **HTML**, **CSS** y **JavaScript**. Todas ellas son interpretadas por el navegador, ultimo elemento que se relaciona con el usuario web.

Los desarrollos web implementados pueden ser **validados** para comprobar su adecuación a los estándares. Entre ellos, tenemos:

- **HTML**: https://validator.w3.org/
- **CSS**: https://validator.w3.org/unicorn/?ucn_task=full-css&ucn_lang=es

Por último, cabe destacar, por su importancia, que los desarrollos webs actuales han de poder ser ejecutados en diferentes plataformas (sistemas operativos) y en diferentes dispositivos (equipos de sobremesa, tablets y móviles), lo que se denomina **Responsive Web Design** (**RWD**) o diseño web adaptable.

BACK-END

El desarrollo de aplicaciones web back-end hace referencia a las tecnologías web que se ubican en el servidor web, acunadas como "**del lado del servidor**". El objetivo básico es recibir las peticiones del cliente (front-end), procesarlas y elaborar una respuesta dirigida al usuario. El back-end, normalmente, se relaciona con diversas fuentes de datos (SGBD principalmente) para el tratamiento de datos.

Las tecnologias web principales del lado del servidor son: **servlets, JSP, ASP.NET, Node.js, PHP, Ruby y Python**.

STACKS DISPONIBLES

Un **stack** es un conjunto de tecnologías (tanto del lado del cliente como del servidor) usadas para construir aplicaciones web. Destacan los siguientes stacks:

- **MEAN**: **M**ongo DB, **E**xpressJS, **Angular** y **N**odeJS.

- **MERN**: **M**ongo DB, **E**xpressJS, **React** y **N**odeJS.

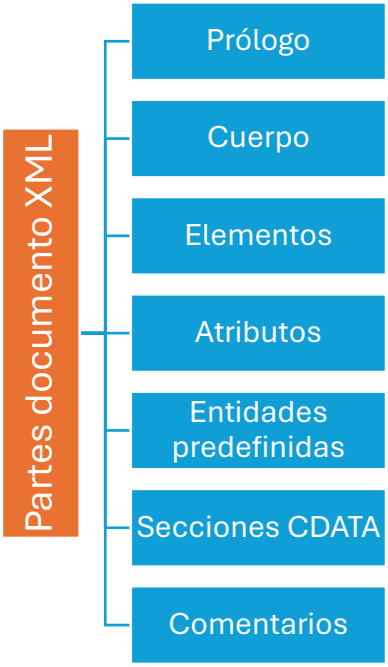
- **MEVN**: **M**ongo DB, **E**xpressJS, **Vue.js** y **N**odeJS.

HTML (HYPERTEXT MARKUP LANGUAGE)

Permite definir la **ESTRUCTURA** de una página web utilizamos HTML. Ver documento **HTML Julio 2024**

XML (EXTENSIBLE MARKUP LANGUAGE)

XML (eXtensible Markup Language) es un lenguaje de marcado que permite describir contenido de forma independiente a su presentación. Estandarizado por el W3C. La última versión es XML 1.1. Deriva del lenguaje SGML (Standard Generalized Markup Language). Las etiquetas XML son case sensitive.



Partes de un documento XML:

1. **PRÓLOGO**: es opcional y contiene una sentencia que declara el documento como XML, una declaración del tipo de documento y comentarios e instrucciones de procesamiento.

   <?xml version="1.0" ?> DOCUMENTO XML

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

   <?xml version="1.0" standalone="no" ?> DOCUMENTO XML

   standalone indica si el documento va acompanado de un DTD (“no”) o no lo necesita

   (“yes”).

   <!DOCTYPE nodoraiz SYSTEM "fichero.dtd"> TIPO DOCUMENTO

   fichero.dtd contiene las reglas (elementos) que se pueden usar en el fichero XML.

   <!DOCTYPE libro SYSTEM "libro.dtd">

2. **CUERPO**: es obligatorio, debe contener solo un elemento raiz (imprescindible para que el documento este **bien formado**).

   <?xml version="1.0" ?>

   <!DOCTYPE libro SYSTEM "libro.dtd">

   <raiz>

   …

   </raiz>

   Si se utiliza un XML Schema para validar el documento XML en lugar de un DTD, se

   añadirán al elemento raiz los siguientes atributos:

   <raiz **xmlns:xs**="http://www.w3.org/2001/XMLSchema-instance"

   xs:schemaLocation="esquema.xsd">

   ...</raiz>

   xs:schemaLocation indica la ubicación del fichero XSD.

3. **ELEMENTOS**: pueden tener contenido o ser elementos vacios.

   <elemento>contenido**</elemento>**

   <elementovacio />

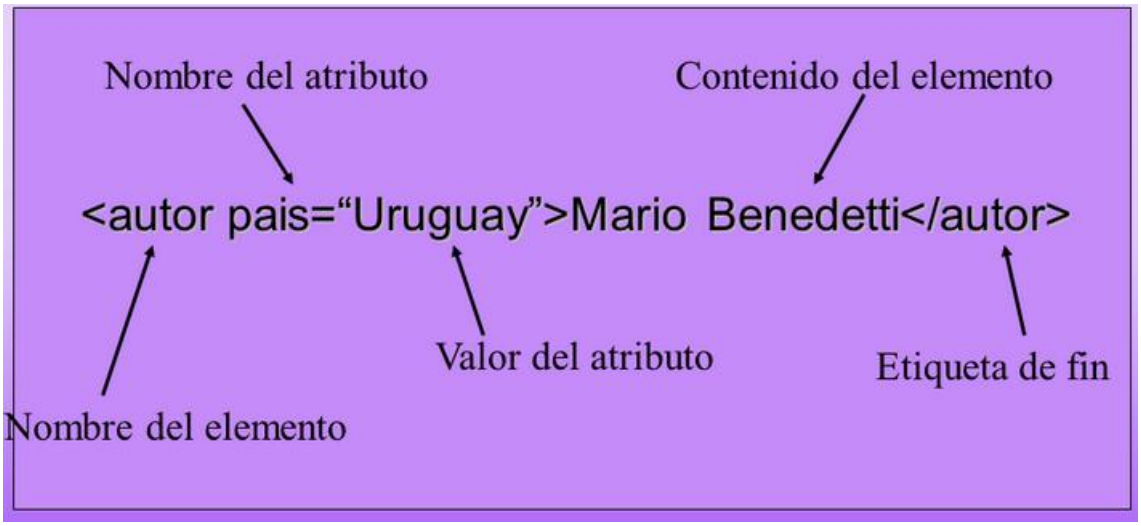
   <nota>8</nota>

   <identificador referencia="23123244"/>

   <linea-horizontal/>

4. **ATRIBUTOS**: permiten incorporar propiedades a los elementos de un documento. Deben ir entre comillas dobles.

   <elemento atributo="valor">



5. **ENTIDADES PREDEFINIDAS**: representan caracteres especiales (por ejemplo, &, <, >, "), de manera que no sean interpretados como elementos de marcado por el procesador XML.

   < menor que

   > mayor que

   & ampensard

   " comilla doble

   ' comilla simple

6. **Secciones CDATA**: permiten especificar datos usando cualquier caracter sin que se interprete como elementos de marcado XML.

   <![CDATA[info]]>

   <ejemplo>

   <![CDATA[

   <HTML>

   <HEAD><TITLE>LISTADO DE APROBADOS</TITLE></HEAD>

   ]]>

   </ejemplo>

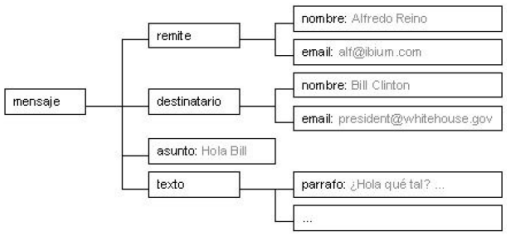
7. **COMENTARIOS**: son ignorados por el procesador XML.

   <!-- Esto es un comentario -->

   Ejemplos:

```
<?xml version="1.0" ?>
<!DOCTYPE libro SYSTEM "libro.dtd">
<!-- Esto es un comentario -->
<libro>
        <titulo idioma="es">XML Pocket Reference</titulo>
        <editorial>Oreilly</editorial>
        <autores>
                <autor>Robert Eckstein</autor>
                <autor>Michel Casablanca</autor>
        </autores>
</libro>
```

```
<?xml version="1.0" encoding="UTF-7"?>
<!DOCTYPE mensaje SYSTEM "mensaje.dtd">
<mensaje>
        <remite>
                <nombre>Alfredo Reino</nombre>
                <email>alf@ibium.com</email>
        </remite>
        <destinatario>
                <nombre>Bill Clinton</nombre>
                <email>president@whitehouse.gov</email>
        </destinatario>
        <asunto>Hola Bill</asunto>
        <texto>
                <parrafo>¿Hola qué tal? ...</parrafo>
        </texto>
</mensaje>
```



Documentos XML bien formados y validos:

- **Documento XML BIEN FORMADO** (cumplimiento reglas **SINTÁCTICAS**): aquel que

respeta la estructura y sintaxis definidas por la especificación de XML:

1. Debe comenzar con una declaración XML.
2. Debe tener un único elemento raíz.
3. Los elementos deben estar cerrados y correctamente anidados.
4. Los valores de los atributos deben ir entre comillas.
5. Los elementos son case sensitive.

- **Documento XML VÁLIDO** (cumplimiento reglas **SEMÁNTICAS**): documento XML que esta bien formado y es conforme a una DTD o XML Schema.

Nota: para que un documento XML sea válido primero ha de estar bien formado.

tecnologías para la validación de documentos XML:

- **DTD (Document Type Definition)**: describe la estructura y sintaxis de un documento XML, definiendo los tipos de elementos, atributos y entidades permitidas. Puede ubicarse en un fichero externo o estar contenida en el propio XML (opcion no

recomendable). Determina:

1. Los ELEMENTOS que pueden formar parte del XML.

   <!ELEMENT nombre_elemento tipo_elemento>

   tipo_elemento puede ser **#PCDATA** (texto), **EMPTY**.

2. Los ATRIBUTOS que pueden tener esos elementos.

   <!ATTLIST nombre_elemento nombre_atributo tipo_atributo modificador>

   tipo_atributo puede ser **CDATA** (cualquier valor o un valor permitido de una lista) y modificador puede ser **#REQUIRED**, **ID** (valor único en todo el documento), **#FIXED** (valores fijos).

3. La GRAMATICA, es decir, la relación entre los elementos. Modificadores del numero de ocurrencias:
   - ✓ **,** (la coma indica el orden exacto de los elementos y su aparición).
   - ✓ **|** (OR, indica opcionalidad, solo se utiliza uno de los elementos de la lista).
   - ✓ **?** (el elemento aparece 0 o 1 vez).
   - ✓ **+** (elemento aparece 1 o varias veces).
   - ✓ ***** (el elemento aparece 0 o varias veces).

<!ELEMENT libro (titulo, autor+, capitulo+, apartados*, CD?)>

<!ATTLIST autor sexo (hombre | mujer) "mujer"> (DEFINE UN ATRIBUTO)

Ejemplo:

<!ELEMENT libro (titulo, editorial, autores)>

<!ELEMENT titulo (#PCDATA)>

<!ATTLIST titulo idioma CDATA>

<!ELEMENT editorial (#PCDATA)>

<!ELEMENT autores (autor+)>

<!ELEMENT autor (#PCDATA)>

- **XML Schemas**: es un documento XML que define los elementos y atributos que puede contener un documento XML. A diferencia de los DTD, se basan en sintaxis XML, y son más potentes al permitir definir nuevos tipos de datos, soportar herencia entre tipos de datos y permitir el uso de namespaces (concepto similar a los paquetes de Java). Esta tecnología también se denomina XSD (XML Schema Definition), siendo una recomendación del W3C.

1. Prologo: presente en todo XML Schema.

   <?xml version="1.0" ?>

2. Elemento raiz (**xs:schema**): presente en todo XML Schema.

   <xs:schema>contenido</xs:schema>

   En el elemento schema, el atributo **xmlns:xs**=http://www.w3.org/2001/XMLSchema indica el espacio de nombres al que pertenecen los elementos y los tipos de datos.

   <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">

3. Elementos simples (**xs:element**): es aquel que solo puede contener texto, del tipo de dato indicado.

   <xs:element name="nombre_del_elemento" type="tipo_de_dato"/>

   tipo_de_dato puede ser, entre otros: xs:string, xs:integer, xs:int, xs:decimal,

   xs:boolean, xs:date, xs:time, xs:dateTime, Y tambien algun tipo definido por el

   usuario.

   Ejemplo:

   <xs:element name="nombre" type="xs:string"/>

   <xs:element name="edad" type="xs:integer"/>

   <xs:element name="fecha_de_nacimiento" type="xs:date"/>

   <xs:element name="hora" type="xs:time"/>

<xs:element name="nota" type="xs:decimal"/>

<xs:element name="apto" type="xs:boolean"/>

<nombre>Elsa</nombre>

<edad>23</edad>

4. Atributos (**xs:attribute**): se ha de indicar el tipo de dato.

   <xs:attribute name="nombre_del_atributo" type="tipo_de_dato"/>

   Los atributos pueden ser **fixed** (fijos), **default** (valor por defecto), **required** y **optional** (si no se indica nada).

   Ejemplo:

   <xs:element name="curso" type="xs:integer"/>

   <xs:attribute name="grupo" type="xs:string"/>

   <xs:attribute name="grupo" type="xs:string" **fixed**="B"/>

   <xs:attribute name="grupo" type="xs:string" **default**="B"/>

   <xs:attribute name="grupo" type="xs:string" **use**="required"/>

   <curso grupo="B">2</curso>

Opción	¿Se puede omitir?	¿Valor aceptado?	¿Valor por defecto?	¿Valor fijo?
`default="B"`	✅ Opcional	Cualquiera	Sí (`"B"`)	❌
`fixed="B"`	✅ Opcional	Solo `"B"`	Sí (`"B"`)	✅
`use="required"`	❌ Obligatorio	Cualquiera	❌	❌

5. Elementos complejos (**xs:complexType**): son aquellos que contienen uno o mas elementos y/o atributos, y también los elementos vacíos.
6. Restricciones (**xs:restriction**): también denominadas facets, permiten definir que valores de los tipos de datos son válidos, tanto para atributos como para elementos.

   xs:length - Longitud fija.

   xs:minLength - Longitud minima.

   xs:maxLength - Longitud maxima.

   xs:pattern - Patron de caracteres admitidos.

   xs:enumeration - Lista de valores admitidos.

   xs:maxInclusive - Valor ≤ que el indicado.

   xs:maxExclusive - Valor < que el indicado.

   xs:minExclusive - Valor > que el indicado.

xs:minInclusive - Valor ≥ que el indicado.

xs:totalDigits - No maximo de digitos para un numero.

xs:fractionDigits - No maximo de decimales para un numero.

xs:simpleType sirve para definir restricciones a un elemento o a un atributo.

Ejemplo: se define un elemento llamado "mes" con la restriccion de que el valor que tome no pueda ser menor que 1 ni mayor que 12.

```
<xs:element name="mes">
    <xs:simpleType>
        <xs:restriction base="xs:integer">
            <xs:minInclusive value="1"/>
            <xs:maxInclusive value="12"/>
        </xs:restriction>
    </xs:simpleType>
</xs:element>
```

Ejemplo: se define un elemento llamado "color" con la restricción de que los únicos

valores admitidos son "verde", "amarillo" y "rojo".

```
<xs:element name="color">
    <xs:simpleType>
        <xs:restriction base="xs:string">
            <xs:enumeration value="verde"/>
            <xs:enumeration value="amarillo"/>
            <xs:enumeration value="rojo"/>
        </xs:restriction>
    </xs:simpleType>
</xs:element>
```

Ejemplo: se define un elemento llamado "letra" con la restricción de que el único valor admitido es una de las letras minúsculas de la "a" a la "z".

```
    <xs:element name="letra">
        <xs:simpleType>
            <xs:restriction base="xs:string">
                <xs:pattern value="[a-z]"/>
```

```
        </xs:restriction>
    </xs:simpleType>
</xs:element>
```

Ejemplo: se define un elemento llamado "clave" con la restricción de que su valor tiene que ser una cadena de exactamente 12 caracteres.

```
<xs:element name="clave">
    <xs:simpleType>
        <xs:restriction base="xs:string">
            <xs:length value="12"/>
        </xs:restriction>
    </xs:simpleType>
</xs:element>
```

7. Indicadores: permiten establecer como se van a utilizar los elementos en un documento XML.
   - ❖ Indicadores de ORDEN:

   xs:sequence: especifica el orden en el que obligatoriamente deben aparecer los elementos hijo de un elemento (AND).

   xs:choice: indica que solo puede aparecer uno de los elementos (OR).

   xs:all: indica que los elementos pueden aparecer en cualquier orden.

   Ejemplo:

```
<xs:element name="lugar">
    <xs:complexType>
        <xs:sequence>
            <xs:element name="ciudad">
                <xs:complexType>
                    <xs:all>
                        <xs:element
                        name="nombre"
                        type="xs:string"/>
                        <xs:element name="pais"
                        type="xs:string"/>
                    </xs:all>
                </xs:complexType>
            </xs:element>
```

</xs:sequence>

</xs:complexType>

</xs:element>

EJEMPLO QUE CUMPLE CON ESE ESQUEMA

<lugar>

<ciudad>

<pais>Italia</pais>

<nombre>Florencia</nombre>

</ciudad>

</lugar>

- § Indicadores de OCURRENCIA: **minOccurs** y **maxOccurs** permiten establecer, respectivamente, el no mínimo y máximo de veces que puede aparecer un determinado elemento. El valor por defecto para maxOccurs y minOccurs es 1.

  Ejemplo:

  <xs:element name="nombre" type="xs:string"/>

  <xs:element name="pais" **maxOccurs="unbounded"**>

  <xs:element name="ciudad" type="xs:string" **minOccurs="0" maxOccurs="5"**/>

- § Indicadores de GRUPO: **xs:group** sirve para agrupar un conjunto de declaraciones de elementos relacionados.

  Ejemplo:

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

  <xs:schema
  xmlns:xs="http://www.w3.org/2001/XMLSchema">

  <xs:element name="personas">

<xs:complexType>

<xs:sequence>

<xs:element name="persona" type="**datosDePersona**"

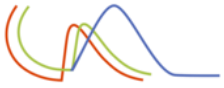
maxOccurs="unbounded"/>

</xs:sequence>

</xs:complexType>

</xs:element>

<xs:complexType name="**datosDePersona**">



```
                    <xs:sequence>
                        <xs:group ref="datosBasicos"/>
                        <xs:element name="telefono" type="xs:string"/>
                    </xs:sequence>
                </xs:complexType>
                <xs:group name="datosBasicos">
                <xs:sequence>
                    <xs:element name="nombre" type="xs:string"/>
                    <xs:element name="edad" type="xs:positiveInteger"/>
                    <xs:element name="pais" type="xs:string"/>
                </xs:sequence>
            </xs:group>
        </xs:schema>
```

Ejemplo: XML Schema

```
<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/21./XMLSchema">
 <xs:element name="libro">
  <xs:complexType>
   <xs:sequence>
    <xs:element name="titulo" type="xs:string"/>
    <xs:element name="editorial" type="xs:string"/>
    <xs:element name="autores">
     <xs:complexType>
      <xs:sequence>
       <xs:element name="autor" type="xs:string" minOccurs="1" maxOc- curs="unbounded"/>
      </xs:sequence>
     </xs:complexType>
    </xs:element>
   </xs:sequence>
  </xs:complexType>
 </xs:element>
</xs:schema>
```

Ejemplo: XML Schema

```
<xs:schema attributeFormDefault="unqualified" elementFormDefault="qualified" xmlns:xs="http://www.w3.org/2001/XMLSchema">
<xs:element name="fnc">
    <xs:complexType>
        <xs:sequence>
```

```
                    <xs:element type="xs:long" name="hash"/>
                    <xs:element type="xs:string" name="userName"/>
                    <xs:element type="xs:short" name="appId"/>
                    <xs:element type="xs:short" name="signId"/>
                </xs:sequence>
            </xs:complexType>
        </xs:element>
        </xs:schema>
```

Dado el esquema XSD anterior, tenemos el siguiente ejemplo de XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<fnc>
        <hash>1628680368248</hash>
        <userName>1R</userName>
        <appId>261</appId>
        <signId>200</signId>
</fnc>
```

Ejemplo: XML Schema

```
<xs:schema attributeFormDefault="unqualified"
elementFormDefault="qualified"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
<xs:element name="fnc">
        <xs:complexType>
            <xs:sequence>
                    <xs:element type="xs:string" name="fileName"/>
                    <xs:element type="xs:string" name="file"/>
                    <xs:element type="xs:string" name="ext"/>
                    <xs:element type="xs:string" name="userName"/>
                    <xs:element type="xs:short" name="appId"/>
                    <xs:element type="xs:long" name="hash"/>
                    <xs:element type="xs:string" name="csv"/>
            </xs:sequence>
        </xs:complexType>
</xs:element>
```

```
</xs:schema>
```

Dado el esquema XSD anterior, tenemos el siguiente ejemplo de XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<fnc>
	<hash>1628680368248</hash>
	<userName>1R</userName>
	<appId>261</appId>
	<signId>200</signId>
</fnc>
```

Ejemplo: XML Schema

```
<xs:schema attributeFormDefault="unqualified" elementFormDefault="qualified"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
<xs:element name="fnc">
	<xs:complexType>
		<xs:sequence>
			<xs:element type="xs:string" name="fileName"/>
			<xs:element type="xs:string" name="file"/>
			<xs:element type="xs:string" name="ext"/>
			<xs:element type="xs:string" name="userName"/>
			<xs:element type="xs:short" name="appId"/>
			<xs:element type="xs:long" name="hash"/>
			<xs:element type="xs:string" name="csv"/>
		</xs:sequence>
	</xs:complexType>
</xs:element>
</xs:schema>
```

Dado el esquema XSD anterior, tenemos el siguiente ejemplo de XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<fnc>
	<fileName>nombreFichero.pdf</fileName>
	<file>JVBERi0xLjQKJeLjz9MKMy[….]</file>
	<ext>.PDF</ext>
```

<userName>1R</userName>

<appId>261</appId>

<hash>1628680368248</hash>

<csv>asiscsv_1628680368450</csv>

</fnc>

PROCESAMIENTO DE DOCUMENTOS XML:

- **DOM (Document Object Model)**: es una API que permite representar objetos de un documento HTML o XML de forma jerárquica. El parser procesa el documento XML completo y elabora un árbol en memoria del mismo, permitiendo a la aplicación recorrer y modificar el árbol. Inconveniente: consume mucha memoria.

- **SAX (Simple API for XML)**: el parser va procesando el documento XML elemento a elemento. Es rápido, pero no permite modificar los datos.

XSL (eXtensible Stylesheet Language)

XSL (eXtensible Stylesheet Language) es un lenguaje que permite definir el formato de un documento XML para su presentación, de manera análoga a lo que ocurre con CSS y HTML.

Es un conjunto de estándares del W3C:

1. **XSLT (XSL Transformations)**: plantillas con una serie de reglas de transformacion y formato que se disparan cuando los elementos coinciden con la plantilla. Transforma un documento XML en otro formato (PDF, HTML...).
2. **XPath (XML Path Language)**: lenguaje que permite construir expresiones para recorrer y procesar un documento XML.
3. **XSL-FO (XSL Formatting Objects)**: es un documento XML que especifica como formatear unos datos para su presentación.
4. **XQuery**: lenguaje de consulta que permite acceder a los datos contenidos en un documento XML.
5. **XLink**: lenguaje que permite crear elementos XML que describen relaciones cruzadas entre documentos, imágenes y archivos.
6. **XPointer**: lenguaje que permite identificar de forma única fragmentos de un documento XML. Esta construido sobre XPath.

LENGUAJES BASADOS EN XML

La sindicación web o sindicación de contenidos es el reenvío de contenidos desde una fuente original (sitio web origen) a un sitio web de destino, habitualmente codificados en XML.

Lenguajes:

- **RSS (Really Simple Syndication)**: formato XML para distribuir contenido en la web.

- **RDF (Resource Description Framework)**: es un modelo estándar para intercambio de datos en la web, siendo una de las tecnologías esenciales de la web semántica. El modelo de datos RDF se basa en hacer declaraciones sobre los recursos en forma de expresiones

sujeto-predicado-objeto (tripleta RDF).

Otros lenguajes basados en XML, relacionados con información financiera:

- **XBRL (eXtensible Business Reporting Language)**: formato estándar basado en XML para intercambio de información contable, financiera y tributaria.

XOP (XML-BINARY OPTIMIZED PACKAGING)

XML-binary Optimized Packaging (**XOP**) es un estandar del W3C que permite serializar de manera más eficiente los conjuntos de informacion XML (infosets XML) que tienen ciertos tipos de contenido.

Los siguientes terminos se utilizan en esta especificación:

- **Orginal XML Infoset**: un conjunto de información XML que se debe optimizar.

- **Optimized Content**: contenido que se ha eliminado del conjunto de información XML.

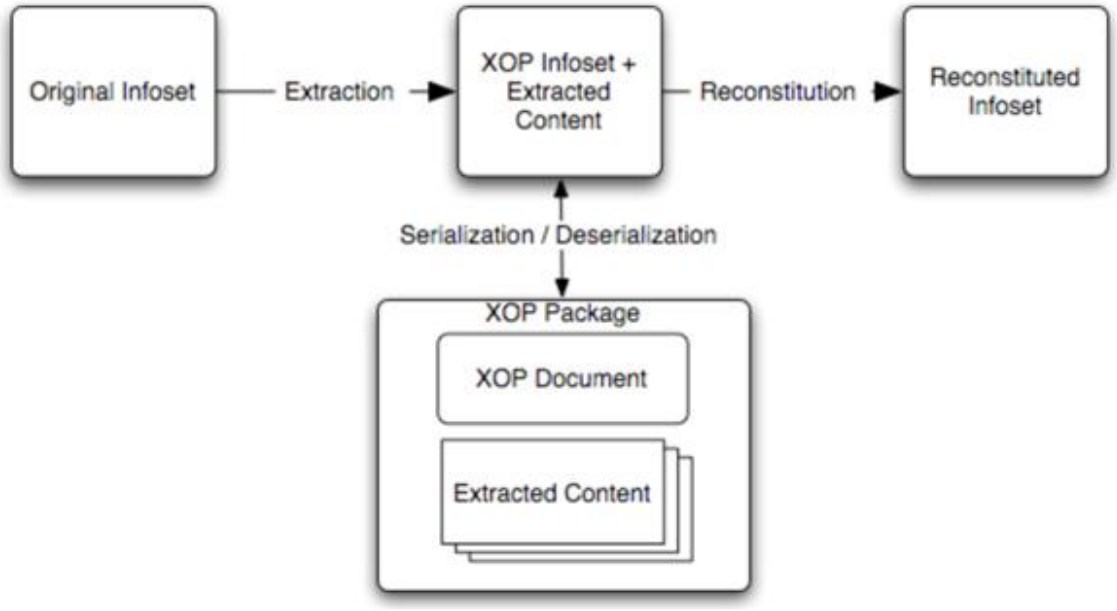
- **XOP Infoset**: Original Infoset + Optimized Content reemplazado por elementos de informacion **xop:Include**.

- **XOP Document**: una serialización del conjunto de informacion XOP utilizando cualquier version de XML.

- **XOP Package**: un paquete que contiene el documento XOP y cualquier contenido optimizado.

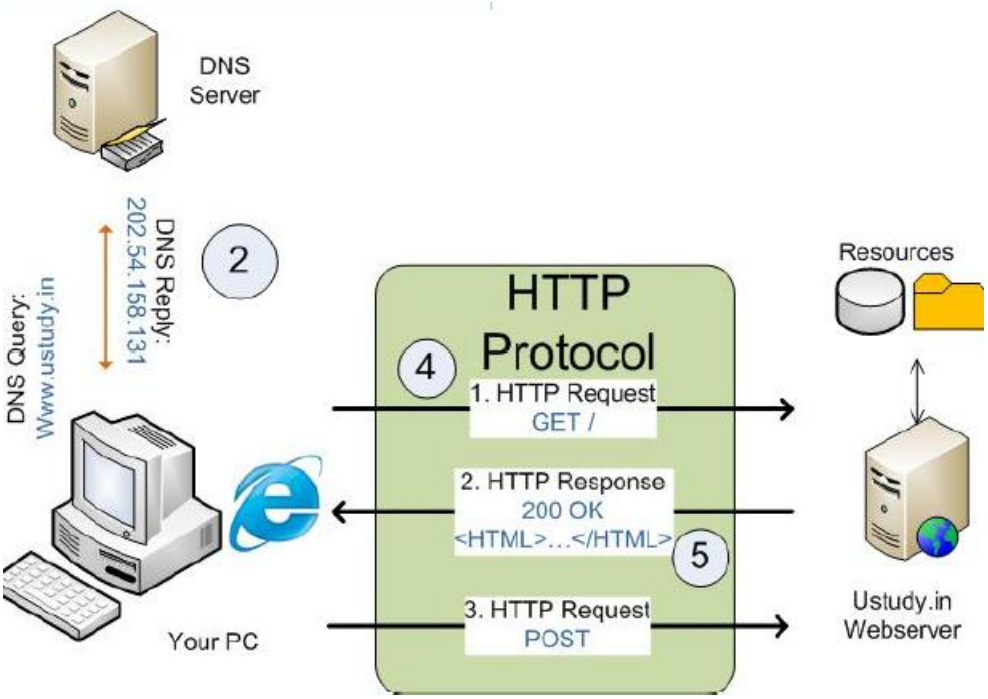
- **Reconstituted XML Infoset**: un conjunto de informacion XML que se ha construido a partir de las partes de un paquete XOP.

MTOM es normalmente usado con XOP.



NAVEGADORES WEB

Un **navegador web** es un programa informático que permite consultar y visualiza páginas web. El navegador interpreta el código, HTML generalmente, en el que está escrita la pagina web y lo presenta en pantalla permitiendo al usuario interactuar con su contenido y navegar. Es de vital importancia conocer los distintos navegadores con los que los usuarios van a utilizar nuestras páginas. En teoría, los estándares web publicados por el W3C deberían permitir que las páginas fueran visualizadas exactamente igual en todos los navegadores. La realidad, sin embargo, es distinta. Cada navegador posee diferencias que pueden hacer necesario el uso de técnicas "especiales" para que una web se muestre de la misma forma en todos los navegadores.



La comunicación entre servidor web y navegador web se realiza a través del protocolo HTTP, aunque cualquier navegador soporta otros protocolos como HTTPS o FTP. Podemos observar un esquema de funcionamiento básico del navegador:

Componentes principales de un navegador web:

- **Interfaz de usuario**: parte visual que interactúa con el usuario. Es la ventana/pestana del navegador.

- **Motor de renderizado**: dibuja el contenido en una ventana/pestana del navegador, que es mostrada al usuario. Esto es, el contenido HTML de una página junto con las CSS es interpretadas por el navegador para generar el contenido visual que ofrecerá al usuario. Destacamos los motores Blink, Webkit, Gecko y Trident.

- **Intérprete JavaScript**: interpreta y ejecuta el código escrito en el lenguaje de programación JavaScript. Entre los interpretes más conocidos citamos a V8, JavaScriptCore y SpiderMonkey.

Los dos últimos componentes determinan el rendimiento de un navegador respecto a la velocidad de carga de una página web.

A continuacion, relacionamos los navegadores web mas relevantes:

(1) Google Chrome: navegador multiplataforma derivado de Chromium.

Motor de renderizado: Blink.

Interprete de JavaScript: V8.

(2) Safari: navegador para dispositivos Apple.

Motor de renderizado: Webkit.

Interprete de JavaScript: JavaScriptCore.

(3) Mozilla Firefox: navegador de codigo abierto y multiplataforma.

Motor de renderizado: Gecko.

Interprete de JavaScript: SpiderMonkey.

(4) Microsoft Edge: navegador derivado de Chromium para Windows, sistemas Apple y Android.

Motor de renderizado: Blink.

Interprete de JavaScript: V8.

(5) Internet Explorer: navegador para Windows. Obsoleto a favor de Microsoft Edge.

Motor de renderizado: Trident.

Interprete de JavaScript: Chakra.

(6) Opera: navegador multiplataforma.

Motor de renderizado: Blink.

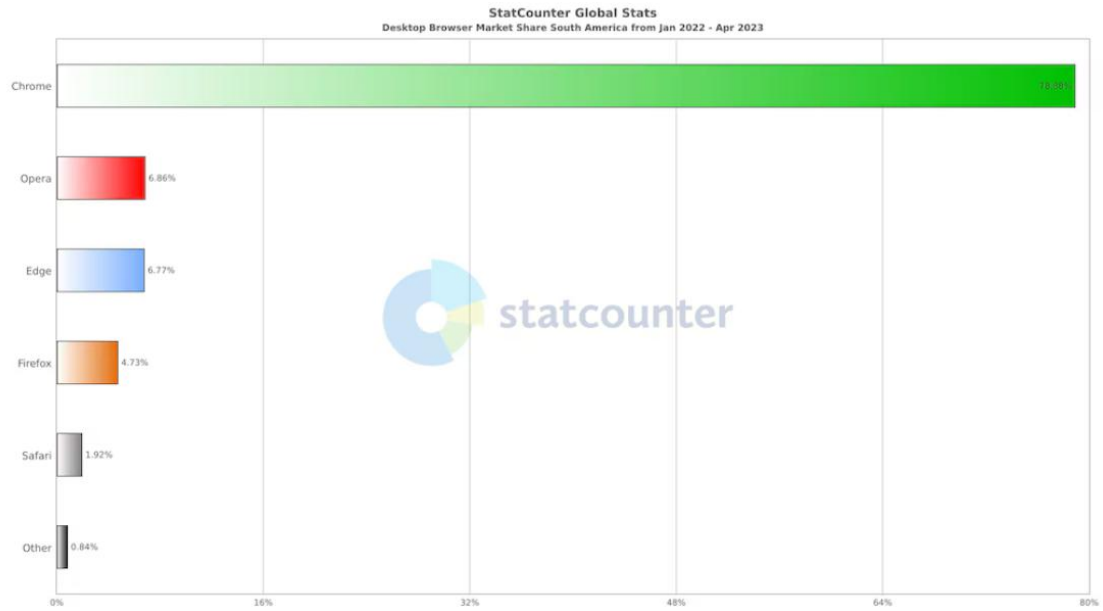
Interprete de JavaScript: V8.

(7) Chromium: navegador de codigo abierto.

(8) Lynx: navegador basado en texto bajo licencia GNU GPL. Se integra con lectores de pantalla destinado a usuarios con problemas visuales. Utilizado para pruebas de

usabilidad de webs en navegadores antiguos.

Estadísticas de uso de navegadores web a nivel mundial en julio de 2022:



Herramientas de comprobación del cumplimiento de los estándares para navegadores web:

- **acid3**: test dirigido a pruebas de DOM y ECMAScript consistente en 100 pruebas encuadradas en 6 grupos (http://acid3.acidtests.org/). La puntuación máxima que se puede obtener es 100.

- **html5test**: evaluación del estándar HTML5 (https://html5test.com). La puntuación máxima que se puede obtener es de 555 puntos.

LENGUAJES DE PROGRAMACIÓN WEB

De entre los lenguajes de programación web o lenguajes del lado del servidor, destacan:

- **PHP**.

- **Python**.

- **Ruby**.

- **Servlet y JSP**.

- **ASP.NET**.

LENGUAJES DE SCRIPT

Los lenguajes de script definen el **COMPORTAMIENTO** de una página web, destacando JavaScript.