Unidad 7.

DOCUMENTOS XML.

Descripción y definición de documentos XML. Validación.

RA4. Establece mecanismos de validación para documentos XML utilizando métodos para definir su sintaxis y estructura.

Criterios de evaluación:

- *a) Se ha establecido la necesidad de describir la información transmitida en los documentos XML y sus reglas.*
- *b) Se han identificado las tecnologías relacionadas con la definición de documentos XML.*
- *c) Se ha analizado la estructura y sintaxis específica utilizada en la descripción.*
- *d) Se han creado descripciones de documentos XML.*
- *e) Se han utilizado descripciones en la elaboración y validación de documentos XML.*
- *f) Se han asociado las descripciones con los documentos.*
- *g) Se han utilizado herramientas específicas.*
- *h) Se han documentado las descripciones*

Recursos
Introducción.
Características.
Estructura de un documento XML.
Validación de documentos XML.
    DTD.
    XSD.
Actividades.

Recursos

Introducción a XML. MDN Web Docs
XML W3C
XML en w3schools
Recursos en xml.com
Tutorial tecnologías XML

Introducción

XML es un lenguaje de marcado similar a HTML. Significa Extensible Markup Language (Lenguaje de Marcado Extensible) y es una **especificación de W3C** como lenguaje de marcado de propósito general. Esto significa que, a diferencia de otros lenguajes de marcado, XML no está predefinido, por lo que **se pueden definir etiquetas propias**. El propósito principal del lenguaje es definir un estándar para describir datos y para poder compartir información entre sistemas diferentes.

XML sirve para estructurar, almacenar e intercambiar información.

Ejemplo de lenguajes basados en XML son XHTML, MathML, SVG, XUL, XBL, RSS, y RDF.

Características.

1 XML permite estructurar datos.

XML es un conjunto de reglas o normas para describir y estructurar datos. Facilita a la generación de documentos para su tratamiento asegurando que su estructura no sea ambigua. XML no es un lenguaje de programación, es un lenguaje extensible, independiente de la plataforma, que soporta internacionalización y localización. Permite incluir datos estructurados como planillas de cálculo, libretas de direcciones, parámetros de configuración, transacciones financieras, dibujos técnicos, …

2 XML, lenguaje de marcado.

Al igual que HTML, XML utiliza elementos similares a los utilizados en HTML como etiquetas y atributos. Mientras HTML especifica lo que cada etiqueta y atributo significa, XML usa las etiquetas sólo para organizar los datos y deja la interpretación de los mismos a las aplicaciones que los manejan. En XML no podemos asumir que <p> se utiliza para delimitar un párrafo, dependiendo del contexto, podría ser un precio, un parámetro, una persona, una palabra, ...

3 XML, texto plano.

Igual que los de HTML, los archivos de XML son archivos de texto y aunque no están pensados para su lectura por desarrolladores y usuarios, este formato facilita mucho el desarrollo y mantenimiento de las aplicaciones.

4 XML, estricto.

Las reglas de XML son estrictas, y en esto se parece menos a HTML. Una etiqueta olvidada o un atributo sin comillas inutilizan un archivo XML, mientras que en HTML es tolerada y a menudo explícitamente permitida. La especificación oficial de XML prohibe a las aplicaciones que traten de adivinar las intenciones del creador de un archivo XML dañado; si el archivo está dañado, la aplicación debe detenerse allí mismo y reportar un error.

5 XML, verboso.

Como XML es un formato de texto y usa etiquetas para delimitar los datos, los archivos XML son casi siempre más grandes que los formatos binarios comparables. Eso fue una decisión consciente de los diseñadores de XML. Las ventajas de un formato de texto son evidentes y las desventajas pueden usualmente ser compensadas en un nivel diferente. El coste del espacio en disco se ha reducido considerablemente y aplicaciones de compresión y protocolos de comunicaciones pueden comprimir los datos de manera muy efectiva para el aprovechamiento del ancho de banda.

6 XML, familia de tecnologías

XML 1.0 es la especificación que define lo que son las "etiquetas" y los "atributos". Más allá de XML 1.0, "la familia XML" es un conjunto creciente de módulos que ofrecen servicios útiles para realizar tareas importantes frecuentemente demandadas. Xlink describe un modo standard de agregar hipervínculos a un archivo XML. XPointer y Xfragments son sintaxis en desarrollo para apuntar a partes de un documento XML. XSL es el lenguaje avanzado para expresar las hojas de estilo. Se basa en XSLT, un lenguaje de transformación usado para reacomodar, agregar y eliminar etiquetas y atributos. El DOM es un conjunto standard de llamadas a funciones para manipular archivos XML (y HTML) desde un lenguaje de programación. XML Schemas 1 y 2 ayudan a los desarrolladores a definir con precisión las estructuras de sus propios formatos basados en XML.

7 XML, histórico

El desarrollo de XML comenzó 1996 y ha sido una recomendación de la W3C desde febrero de 1998. Antes de XML estuvo SGML, desarrollado a principios de los '80, standard ISO desde 1986, y ampliamente usado para grandes proyectos de documentación. El desarrollo de HTML empezó en 1990. Los diseñadores de XML simplemente tomaron las mejores partes de SGML, guiados por la experiencia con HTML, y produjeron algo que es no menos poderoso que SGML, y vastamente más regular y simple de usar. Algunas evoluciones, sin embargo, son difíciles de distinguir de revoluciones... y hay que decir que mientras SGML es mayormente usado para documentación técnica y mucho menos para otras clases de datos, con XML pasa exactamente lo opuesto.

8 XML lleva HTML a XHTML

Aplicado a HTML lleva al estándar XHTML para su utilización en la web.

9 XML es modular

XML le permite definir un formato de documento combinando y reusando otros formatos. Puesto que dos formatos desarrollados independientemente podrían tener elementos o atributos con el mismo nombre, se debe tener cuidado al combinarlos (¿"<p>" significa "parágrafo" de este formato o "persona" de aquél otro?). Para eliminar la confusión de nombres al combinar formatos, XML provee un mecanismo de espacio de nombre. XSL y RDF son buenos ejemplos de formatos basados en XML que usan espacios de nombres.

9. XML es la base de RDF y de la Web Semántica

Resource Description Framework (RDF) de la W3C es un formato de texto XML que soporta aplicaciones de descripción de recursos y metadatos, tales como listas de temas musicales, colecciones de fotos, y bibliografías. Por ejemplo, RDF podría permitirle identificar las personas en un álbum de fotos Web usando información de una lista de contactos personales; entonces, su cliente de correo podría enviar automáticamente a esas personas un mensaje diciendo que sus fotos están en la Web.

10. XML es gratuito, independiente de la plataforma y bien soportado.

Estructura de un documento XML.

Un documento XML está formado por dos partes: :

- **Prólogo**: Es una parte opcional al principio del documento, que incluye información diversa sobre sus características. Proporciona información sobre el propio documento. Está formado por:
  - Definición XML. Determina la versión XML, codificación y si la validez del documento depende de otro documento externo, con los atributos version, encoding y standlone. Se escribe utilizando como delimitadores los caracteres: *<?xml ?>*

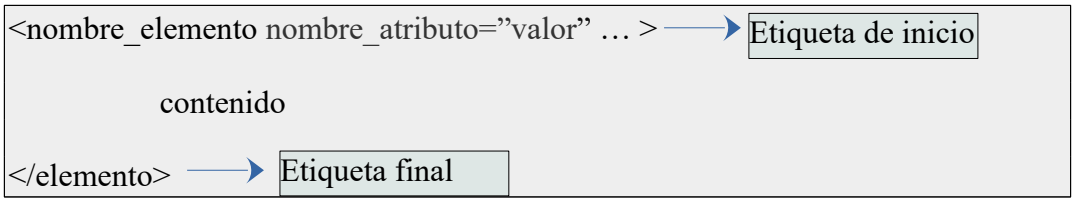
```
<?xml versión= “1.0” ?>
<?xml versión= “1.0” encoding=”utf-8” ?>
<?xml versión= “1.0” encoding=”utf-8” standalone=”yes” ?>
```

  - Declaración del tipo de documento. Define el tipo de documento y las especificaciones necesarias para su correcto procesamiento. Se escribe utilizando como delimitadores: caracteres: *<!DOCTYPE >*

```
<!DOCTYPE nombre_ejemplar ... >
```

- **Ejemplar**: Es la parte que contiene realmente el documento xml. Está formado por un conjunto de elementos xml organizados en una estructura jerárquica, cuyo nodo raíz es el ejemplar. Cada elemento xml está formado por:
  - Etiqueta de inicio. Consta del carácter de apertura < , nombre del elemento xml, opcionalmente una lista de pares atributo:valor y carácter de cierre >.
  - Contenido. Un elemento xml puede contener: texto, elementos xml, una combinación de ambos, o estar vacío.
  - Etiqueta final. Consta del carácter final > y el nombre del elemento precedido del carácter /

  Elemento XML

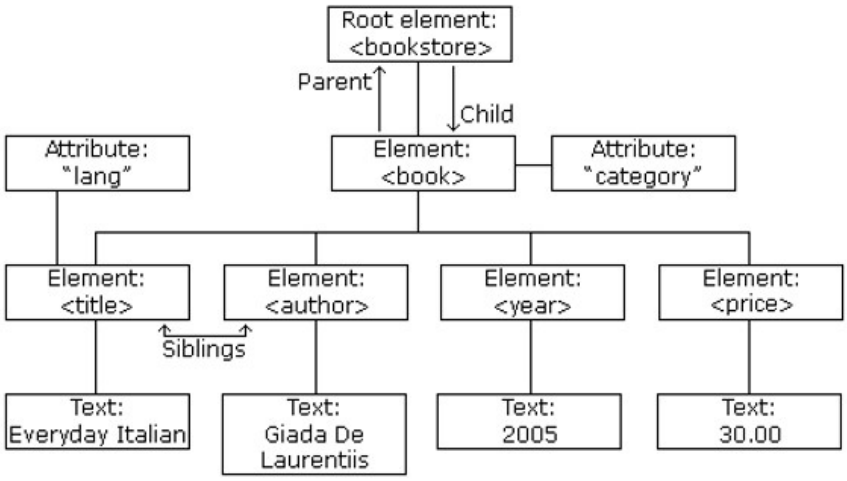


Ejemplo w3schools.

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

<bookstore>

<book category="COOKING">
  <title lang="en">Everyday Italian</title>
  <author>Giada De Laurentiis</author>
  <year>2005</year>
  <price>30.00</price>
</book>

<book category="CHILDREN">
  <title lang="en">Harry Potter</title>
  <author>J K. Rowling</author>
  <year>2005</year>
  <price>29.99</price>
</book>

<book category="WEB">
  <title lang="en">XQuery Kick Start</title>
  <author>James McGovern</author>
  <author>Per Bothner</author>
  <author>Kurt Cagle</author>
  <author>James Linn</author>
  <author>Vaidyanathan Nagarajan</author>
  <year>2003</year>
  <price>49.99</price>
</book>

<book category="WEB">
  <title lang="en">Learning XML</title>
  <author>Erik T. Ray</author>
  <year>2003</year>
  <price>39.95</price>
</book>

</bookstore>
```

- Identifica el prólogo.
- Identifica el ejemplar.
- ¿Cuál es el nodo raíz?
- ¿Qué es *category*?
- Señala un valor.

Árbol


Root element:
<bookstore>
Parent
Child
Attribute:
"lang"
Element:
<book>
Attribute:
"category"
Element:
<title>
Element:
<author>
Element:
<year>
Element:
<price>
Siblings
Text:
Everyday Italian
Text:
Giada De
Laurentiis
Text:
2005
Text:
30.00

Sintaxis de un documento XML.

Las reglas sintácticas que deben cumplir los documentos XML son muy simples y lógicas. Un documento XML diremos que está **bien formado** cuando cumpla estas reglas sintácticas.

- Un documento XML tiene un elemento o nodo raíz que agrupa el resto de componentes del documento.

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

  <nodo>
      ...
  </nodo>
      ...

</nodo_raiz>
```

- Todo elemento XML debe tener una etiqueta de cierre. (La declaración de un documento no es una parte del propio documento y por tanto no tiene etiqueta de cierre).

```
<title lang="en">Las tres bodas de Manolita</title>
```

- Los elementos XML deben estar correctamente anidados.

- Las etiquetas XML son sensibles a mayúsculas y minúsculas.

- El valor de los atributos XML deben aparecer entrecomillados.

```
<book category="WEB">
```

- Se utilizan entidades para referenciar algunos caracteres especiales.

&lt;	<
&gt;	>
&amp;	&
&apos;	'
&quot;	“

- En XML los espacios en blanco no se truncan a diferencia de lo que ocurre en HTML.

- Los comentarios en XML se introducen con los caracteres de marcado <!-- -->

```
<!-- comentario  -->
```

- XML almacena una nueva línea con el carácter LF (line feed).

- Los elementos XML deben seguir las siguientes reglas de nombres:
  - Los nombres pueden contener letras, números y otros caracteres.
  - Los nombres no pueden empezar con un número o un carácter de puntuación.
  - Los nombres no pueden comenzar con las letras xml.
  - Los nombres no pueden contener espacios en blanco.

Buenas prácticas en el uso de nombres son:

- Utilizar nombres descriptivos. ***<primer_apellido>***
- Nombres cortos y simples. Por ejemplo ***<titulo_libro>*** y no ~~***<el_titulo_del_libro>***~~
- Evitar caracteres que puedan provocar ambigüedades.
- Cuando se utilizan bases de datos utilizar las mismas reglas para nombrar los elementos.
- Evitar caracteres que puedan presentar problemas de compatibilidad.

Los elementos XML pueden ser extendidos para incluir más información y los programas que lo interpretan no producen un error por estas extensiones.

Ejemplo

```
<?xml version="1.0" encoding="UTF-8"?>
<nota fecha=”01/01/2012”>
  <para>Mercedes</para>
  <de>Jose</de>
  <cabecera>Recordatorio</cabecera>
  <cuerpo>No olvides el cumpleaños de Andrés</cuerpo>
</nota>
```

Supongamos que disponemos de una aplicación que muestra la información del siguiente modo:

MENSAJE
Para: Mercedes
De: Jose
No olvides el cumpleaños de Andrés

Si posteriormente se añade un nuevo elemento al documento XML, la aplicación sigue mostrando la información correctamente.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<nota fecha="01/01/2012">
  <fecha>01/01/2012</fecha>
  <para>Mercedes</para>
  <de>Jose</de>
  <cabecera>Recordatorio</cabecera>
  <cuerpo>No olvides el cumpleaños de Andrés</cuerpo>
</nota>
```

Atributos

Los atributos aparecen en la etiqueta de inicio, acompañando al nombre del elemento y aportando información adicional sobre éste. Deben aparecer entrecomillados. Se pueden usar comillas simples o dobles. Esto permite incluir comillas en el valor de un atributo.

```xml
<persona sexo="hombre">
<gangster name='George "Shotgun" Ziegler'>
<gangster name="George &quot;Shotgun&quot; Ziegler">
```

El atributo ***id*** se utiliza para referenciar e identificar elementos XML

```xml
<?xml version="1.0"?>
<mensajes>
  <nota id="100">
    <para>Mercedes</para>
    <de>Jose</de>
    <cabecera>Recordatorio</cabecera>
    <cuerpo>No olvides el cumpleaños de Andrés</cuerpo>
  </nota>
  <nota id="101">
    <para>Jose</para>
    <de>Mercedes</de>
    <cabecera>Recordatorio</cabecera>
    <cuerpo>Ya compré el regalo</cuerpo>
  </nota>
</mensajes>
```

El atributo id permite identificar las diferentes notas. No es una parte propia de la nota. El metadato (datos sobre los datos) es almacenado como atributo.

Atributos vs Elementos.

En ocasiones es posible describir la misma información utilizando elementos o atributos. En ambos casos la información almacenada es la misma y no hay criterios generales para aplicar un modelo u otro. En cualquier caso, se pueden considerar las siguientes cuestiones.

- Los atributos no pueden contener valores múltiples.
- Los atributos no pueden contener una estructura de árbol.
- Los atributos son difíciles de extender.
- Los atributos son difíciles de leer y mantener.
- Los elementos aportan mayor flexibilidad y conviene dejar los atributos sólo para información no relevante de los datos.

```
<persona sexo="mujer">
  <nombre>Ana</nombre>
  <apellido>López</apellido>
</persona>

<persona>
  <sexo> mujer</sexo>
  <nombre>Ana</nombre>
  <apellido>López</apellido>
</persona>
```

Actividad 1.

Construir un documento XML para describir la información del siguiente mensaje, recibido por correo electrónico de una ONG:

“Hola necesito conocer la población de una serie de países del planeta. De cada país deseo conocer el nombre y los millones de habitantes. De momento, conocemos que Francia tiene 59,7 millones de habitantes y que Irlanda tinen 3,8 millones de habitantes”.
Una vez enviada la primera versión del documento nos vuelven a mandar un correo indicando lo siguiente:
“Hola de nuevo, gracias por su XML pero necesitaría incluir a Brasil que tiene 176,0 millones de habitantes”

Actividad 2.

Construir un documento XML para el siguiente pedido que nos ha enviado por correo electrónico Jaime Peña:

“Hola, últimamente ando un poco liado con los reyes del planeta. De cada país me gustaría conocer el nombre, la capital y el nombre del rey o de la reina. Hay que tener en cuenta que un país puede tener un rey y también puede tener una reina. De momento necesito los datos de Gran Bretaña, Francia y Noruega. Para saber los nombres D. Jaime Peña nos indica que consultemos Wikipedia.

Actividad 3.

Construir un documento XML para el siguiente pedido que ha recibido por correo electrónico una empresa dedicada a la venta de herramientas para jardinería:

"Hola, necesito una cortadora de césped para mi jardín, me gustaría que fuera uno de esos modelos eléctricos, pues las de gasolina contaminan mucho. Me llamo Roberto Movilla, la cortadora la tendrán que enviar a Albacete, la dirección es Plaza de la Duquesa 12, la recogerá mi esposa que se llama Alicia Abad. Ahora que lo pienso también necesitaré 3 podadoras para los setos. Mi dirección es General Ricardos 56, aquí en Madrid. Es urgente, por favor, el césped está muy alto."
La fecha del pedido es el 20 de octubre del 99. El empleado que se encarga del pedido ha comprobado algunos datos necesarios: el código postal de la dirección de Albacete es 05020 y el de la de Madrid 28055; también ha consultado el catálogo de productos y ha averiguado que la cortadora vale 148.95 euros y su código de producto es 872-AA, aunque no sabe si es o no eléctrica; una podadora vale 7.98 y su código es 926-FH.

Documentos XML bien formados.

Son documentos XML sintácticamente correctos.

- Nodo raíz.
- Cierre de etiquetas.
- Sensible a mayúsculas y minúsculas.
- Correcto anidamiento de etiquetas.
- Atributos entrecomillados.

Documentos XML válidos.

Un documento XML válido es un documento bien formado que cumple un conjunto de especificaciones descritas en un documento referente: DTD, XSD

Validación de documentos XML

Un documento XML válido es un documento bien formado que cumple un conjunto de especificaciones descritas en un documento referente: DTD, XSD

XML DTD (Document Type Definition)

El propósito de un DTD es definir la estructura de un documento XML y que éste debe cumplir para que sea válido.

Ventajas del uso de DTD.

- Cada documento XML puede incluir su propia descripción.
- Grupos independientes pueden usar un DTD estándar para intercambiar datos.
- Nuestras aplicaciones pueden usar un estándar DTD para verificar que los datos que reciben desde el exterior son válidos.
- Un DTD nos puede ayudar para verificar nuestros propios datos.

La definición del documento puede incluirse en el mismo documento XML o en un documento externo. Hay que aclarar que en este segundo caso el documento **DTD no es un documento XM**L.

Declaración interna.

La información DTD se incluye en el mismo fichero XML utilizando la siguiente sintaxis:

```
<!DOCTYPE elemento-raíz [ declaraciones ]>
```

Actividad

- Copiamos el siguiente documento XML con su descripción interna.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE note [
<!ELEMENT note (to,from,heading,body)>
<!ELEMENT to (#PCDATA)>
<!ELEMENT from (#PCDATA)>
<!ELEMENT heading (#PCDATA)>
<!ELEMENT body (#PCDATA)>
]>
<note>
  <to>Mercedes</to>
  <from>Jose</from>
  <heading>Recordar</heading>
  <body>No olvides el cumpleaños de Andrés!</body>
</note>
```

- Comprobamos que el documento está bien formado.
- Comprobamos que el documento es válido.

Declaración externa.

La declaración externa puede ser privada o pública.

Privada: <!***DOCTYPE*** *elemento-raíz* ***SYSTEM*** *"URI">*
Pública: <!***DOCTYPE*** *elemento-raíz* ***PUBLIC*** *"Id-público" "URI">*

- ¿Dónde hemos utilizado una declaración pública?

Actividad.

- Creamos el siguiente documento XML

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE note SYSTEM "nota.dtd">
<note>
  <to>Mercedes</to>
  <from>Jose</from>
  <heading>Recordar</heading>
  <body>No olvides el cumpleaños de Andrés!</body>
</note>
```

- Creamos el siguiente DTD

```
<!ELEMENT note (to,from,heading,body)>
<!ELEMENT to (#PCDATA)>
<!ELEMENT from (#PCDATA)>
<!ELEMENT heading (#PCDATA)>
<!ELEMENT body (#PCDATA)>
```

- Comprobamos que el documento está bien formado.

- Comprobamos que el documento es válido.

La definición de tipo de documento (DTD) se realiza declarando:

- Elementos.
- Atributos.
- Entidades.
- Notaciones.

Elementos.

```
<!ELEMENT nombre-del-elemento tipo-de-contenido>
```

El tipo de contenido puede ser:

- Texto. **#PCDATA** (Parsed Character Data)
- Elementos XML (*lista_nodos_hijos*)
  - Cuando los hijos son declarados en una lista separados por comas, los hijos deben aparecer en el mismo orden en el documento.
  - Declarando una o más ocurrencias de un elemento +

    ```
    <!ELEMENT nombre_elemento (nombre_hijo+)>
    ```

    Ejemplo*: <!ELEMENT note (message+)>*
  - Declarando cero o más ocurrencias de un elemento *

    ```
    <!ELEMENT nombre_elemento (nombre-hijo*)>
    ```

    Ejemplo*: <!ELEMENT nota (mensaje*)>*
  - Declarando cero o una ocurrecia de un elemento **?**

    ```
    <!ELEMENT element-name (child-name?)>
    ```

    Ejemplo*: <!ELEMENT note (message?)>*
  - Declarando uno u otro contenido |

    ```
    <!ELEMENT element-name (h1|h2|h3)>
    ```

    Ejemplo: <!ELEMENT note (to,from,header,(message|body))>
  - Declarando contenido mixto.

    Ejemplo*: <!ELEMENT note (#PCDATA|to|from|header|mensaje)*>*
- Vacío. **EMPTY**
- Contenido mixto. **ANY** (No se recomienda)

Ejemplo 1.

```
<!ELEMENT note (to,from,heading,body)>
<!ELEMENT to (#PCDATA)>
<!ELEMENT from (#PCDATA)>
<!ELEMENT heading (#PCDATA)>
<!ELEMENT body (#PCDATA)>
```

Ejemplo 2.

```
!ELEMENT persona (nombre, mayor_de_edad, ciudad)>
!ELEMENT nombre (#PCDATA)>
!ELEMENT mayor_de_edad EMPTY>
!ELEMENT ciudad (#PCDATA)>
```

Actividad.

- Creamos un documento XML válido para la definición del ejemplo anterior.

Atributos

<!***ATTLIST*** *nombre_elemento nombre_atributo tipo_atributo valor_defecto*>

Los tipos de atributo pueden ser los siguientes:

- CDATA: El valor es un character data.
- (en1|en2|...): El valor debe ser uno de los enumerados.
- ID: El valor es un único id.
- IDREF: El valor es el id de otro elemento.
- IDREFS: El valor es una lista de otros ids.
- NMTOKEN: El valor es un nombre XML válido.
- NMTOKENS: El valor es una lista de nombres XML válidos.
- ENTITY: El valor es una entidad.
- ENTITIES: El valor es una lista de entidades.
- NOTATION: El valor es el nombre de una notación.
- XML: El valor es un valor XML predefinido.

Los valores por defecto pueden ser:

- value Valor por defecto del atributo.
- #REQUIRED El atributo es requerido.
- #IMPLIED El atributo no es requerido.
- #FIXED value El valor del atributo es fijo

Ejemplo 1.

DTD:

```
<!ATLIST forma_pago tipo CDATA “cheque”>
```

XML:

```
<forma_pago tipo=”cheque”>
```

Ejemplo 2. El element cuadrado puede ser un elemento vacío con un atributo de tipo CDATA y si no se especifica la anchura su valor es 0

DTD:

```
<!ELEMENT cuadrado EMPTY>
<!ATTLIST cuadrado width CDATA "0">
```

XML válido:

```
<cuadrado width="100" />
```

Ejemplo 3: Usamos #REQUIRED si no tenemos la opción por defecto pero queremos forzar que el atributo esté presente.

DTD:
<!ATTLIST persona numero CDATA #REQUIRED>
XML válido:
<persona numero="5677" />
XML inválido:
<persona />

Ejemplo 4. Usamos #IMPLIED si no queremos forzar que el autor incluya un atributo y no tenemos una opción de valor por defecto.

DTD:
<!ATTLIST contacto fax CDATA #IMPLIED>
XML válido:
<contacto fax="555-667788" />
XML válido:
<contacto />

Ejemplo 5. Usamos #FIXED cuando queremos que el atributo tenga un valor fijo impidiendo que el usuario pueda cambiarlo.

DTD:
<!ATTLIST remitente empresa CDATA #FIXED "Microsoft">
XML válido:
<remitente empresa="Microsoft" />
XML inválido:
<sender company="IES Gran Capitán" />

Ejemplo 6. Usamos valores enumerados cuando queremos que el atributo tome su valor de un conjunto de valores posibles.

DTD:
<!ATTLIST forma_pago tipo (cheque|metálico) "metálico">
XML válidos:
<forma_pago tipo="cheque" />
<forma_pago tipo="metálico" />

Entidades.

Las entidades son variables usadas para definir acceso rápido a texto estándar o caracteres especiales.

Declaración interna.

<!***ENTITY*** nombre_entidad “valor_entidad”>

Ejemplo.

DTD

```
<!ENTITY escritor "Miguel de Cervantes.">
<!ENTITY copyright "IES Gran Capitán.">
```

XML válido:

```
<autor>&escritor;&copyright;</author>
```

Declaración externa.

<!***ENTITY*** nombre_entidad ***SYSTEM*** "URI/URL">

Ejemplo.

DTD:

```
<!ENTITY escritor SYSTEM "http://www.iesgrancapitan.org/entidades.dtd">
<!ENTITY copyright SYSTEM "http://www.iesgrancapitan.org/entidades.dtd">
```

XML válido:

```
<author>&writer;&copyright;</author>
```

Notaciones

La definición de notaciones se pueden utilizar para especificar el formato de entidades externas (datos no XML), como por ejemplo un archivo que contenga una imagen. Dichas entidades externas no las analizará un procesador XML, sino que serán tratadas por el programa que procese el documento.

En una DTD se pueden declarar dos tipos de notaciones: privadas y públicas. Para las privadas se utiliza SYSTEM, y para las públicas PUBLIC, pudiéndose utilizar las siguientes sintaxis:

```
<!NOTATION nombre-notación SYSTEM "identificador-sistema">
<!NOTATION nombre-notación PUBLIC "identificador-público">
<!NOTATION nombre-notación PUBLIC "identificador-público" "identificador-del-sistema">
```

Ejemplo 1.

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE frutas [
  <!ELEMENT frutas (fruta)*>
  <!ELEMENT fruta EMPTY>
  <!ATTLIST fruta foto ENTITY #REQUIRED>

  <!ENTITY manzana SYSTEM "manzana.gif" NDATA gif>
  <!ENTITY naranja SYSTEM "naranja.gif" NDATA gif>

  <!NOTATION gif SYSTEM "image/gif">
]>

<frutas>
  <fruta foto="manzana"/>
  <fruta foto="naranja"/>
</frutas>
```

Ejemplo 2.

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE frutas [
  <!ELEMENT frutas (fruta)*>
  <!ELEMENT fruta EMPTY>
  <!ATTLIST fruta foto ENTITY #REQUIRED>

  <!ENTITY manzana SYSTEM "manzana.gif" NDATA gif>
  <!ENTITY naranja SYSTEM "naranja.gif" NDATA gif>

  <!NOTATION gif PUBLIC "GIF 1.0">
]>

<frutas>
  <fruta foto="manzana"/>
  <fruta foto="naranja"/>
</frutas>
```

Actividad

- Diseñamos un documento XML para registrar de forma detallada los errores que aparecen en las actividades de la unidad.

Actividad. <ejemplo13.xml>

- Creamos la siguiente definición de elementos.

```
<!ELEMENT etiqueta (nombre, calle, ciudad, pais, codigo)>
<!ELEMENT nombre (#PCDATA)>
<!ELEMENT calle (#PCDATA)>
<!ELEMENT ciudad (#PCDATA)>
<!ELEMENT pais (#PCDATA)>
<!ELEMENT codigo (#PCDATA)>
```

- Creamos el siguiente documento XML

```
<etiqueta>
  <nombre>Pepe García</nombre>
  <calle>C/Ronda, 3</calle>
  <pais>España</pais>
  <codigo>18465</codigo>
</etiqueta>
```

- Con declaración interna comprobamos que el documento está bien formado y es válido.

- Corregimos el documento XML para hacerlo válido, registrando en el fichero de errores los errores encontrados.

Actividad. <ejemplo14.xml>

- Creamos el fichero ejemplo-agenda.dtd con las siguientes declaraciones.

```
<!ELEMENT agenda (persona)+>
<!ELEMENT persona (nombre, tlf)>
<!ATTLIST persona id ID #REQUIRED>
<!ELEMENT nombre (#PCDATA)>
<!ELEMENT tlf (#PCDATA)>
```

- Creamos el siguiente documento XML

```
<!-- Incluir declaración de documento -->
<!-- Declaración externa de dtd -->
<persona>
  <nombre>Ricardo Borriquero</nombre>
  <tlf>951345679
</persona>
<persona id="eva">
  <nombre>Eva Risto</nombre>
  <tlf>955837659<tlf>
  <tw>@evaristo</tw>
</persona>
```

- Corregimos el documento XML para hacerlo válido, registrando en el fichero de errores los errores encontrados.

Actividad. <ejemplo15.xml>

- Creamos la siguiente definición de elementos.

```
<!ELEMENT relacion (persona)+>
<!ELEMENT persona (nombre, email*, parentesco?)>
<!ATTLIST persona carnet ID #REQUIRED>
<!ATTLIST persona sexo (hombre|mujer) #IMPLIED>
<!ELEMENT nombre (#PCDATA)>
<!ELEMENT email (#PCDATA)>
<!ELEMENT parentesco EMPTY>
<!ATTLIST parentesco tutor-legal IDREFS #IMPLIED hermano-a IDREFS #IMPLIED>
```

- Creamos un documento XML válido con la definición anterior.

Actividades propuestas.
Documentos bien formados y validación DTD.

1 Encuentra, registra y corrige los errores de los siguientes documentos XML.

deportistas.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<deportistas>
<deportista>
<deporte Atletismo />
<nombre>Jesse Owens</nombre>
<deportista>
<deporte Natación />
<nombre>Mark Spitz</nombre>
</deportista>
</deportistas>
```

peliculas.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<pelicula>
<titulo>Con faldas y a lo loco</titulo>
<director>Billy Wilder</director>
</pelicula>
<pelicula>
<director>Leo McCarey</director>
<titulo>Sopa de ganso</titulo>
</pelicula>
<autor />barto</autor>
```

texto.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<texto>
<Titulo>XML explicado a los niños</titulo>
<párrafo>El <abreviatura>XML</abreviatura>define cómo crear lenguajes de marcas.
</párrafo>
<párrafo>Las marcas se añaden a un documento de texto para añadir información.</párrafo>
 <http://>www.example.org</http://>
</texto>
```

infgeografica.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<geografia mundial>
 <pais>
 <pais>España</pais>
 <continente>Europa</continente>
 <capital></capital nombre="Madrid">
 </pais>
 </geografia mundial>
```

programas

```
<?xml version="1.0" encoding="UTF-8"?>
<programas>
<programa nombre="Firefox" licencia="GPL" licencia="MPL" />
<programa nombre="LibreOffice" licencia="LGPL" />
<programa nombre="Inkscape" licencia=GPL />
</programas>
```

mundiales_futbol.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<mundiales-de-futbol>
<mundial>
<pais="España" />
<1982 />
</mundial>
</mundiales-de-futbol>
```

medios_transporte

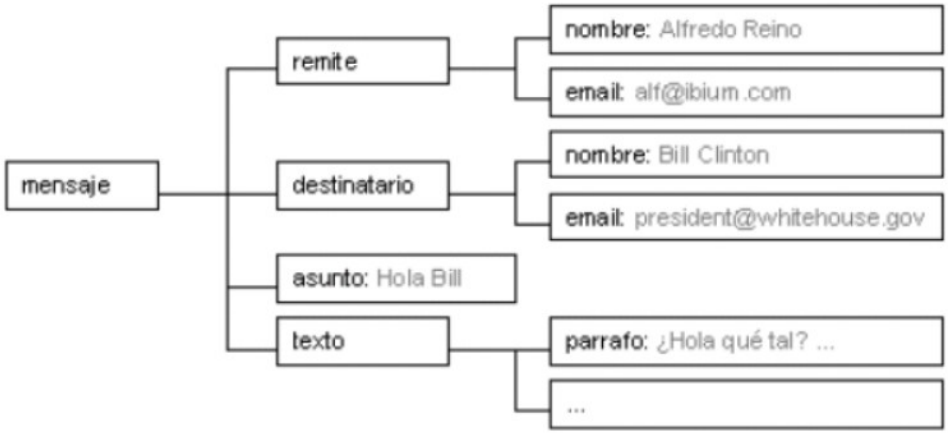
```
<?xml version="1.0" encoding="UTF-8"?>
<mediosDeTransporte>
<bicicleta velocidad="v<100km/h" />
<patinete velocidad maxima="50 km/h"
</mediosDeTransporte>
```

2 Escribe un documentos XML para describir y almacenar la información relevante que aparece en cada uno de los siguientes enunciados.

- Pedido para el señor Juan Delgado Martínez. El pedido se compone de una bicicleta A2023. A entregar en la calle Barco 4, tercer piso, letra A, el día 19-5-2000.

3 Escribe un documento XML que represente la estructura y los datos que se muestran en la imagen.
Completar el contenido del documento para que el texto completo sea el siguiente (tres párrafos en total):

- ¿Hola qué tal?
- Hace mucho tiempo que no escribes. A ver si llamas y quedamos pronto.
- Un saludo.



4 Escribe un documento XML y su representación en forma de árbol, de la información recogida en una carta de desayunos de un restaurante.

Nombre	Precio (en euros)	Descripción	Calorías
Gofres Belgas	5.95	Dos de nuestros famosos Gofres belgas con abundante sirope	650
Gofres Belgas con fresas	7.95	Ligeros gofres belgas cubiertos de fresas y nata montada	900
Gofres Belgas con frutas del bosque	8.95	Ligeros gofres belgas cubiertos con frutas del bosque y nata montada	900
Tostada Francesa	4.50	Dos gruesas rebanadas de nuestro pan francés casero	600
Desayuno de la casa	6.95	Dos huevos, bacon o salchicha, tostada y patatas fritas	950

5 Escribe un documento XML para recoger la siguiente información sobre árboles:

Acer monspessulanum
Nombre común: Arce de Montpellier, Arce menor
Vegetación: Caducifolio
Altura: De 6 a 10 metros
Forma y estructura: Copa esférica. Tronco principal recto con bifurcaciones. Ramaje colgante
Color en primavera: Haz verde brillante, envés verde blanquecino
Resistencia a las heladas: Heladas fuertes (hasta -15ºC)
Olea europea
Nombre común: Olivo
Vegetación: Perenne
Altura: De 8 a 15 metros
Forma y estructura: Copa irregular. Tronco principal irregular con bifurcaciones.
Ramaje tortuoso
Color en primavera: Haz verde oscuro, envés verde plateado
Resistencia a las heladas: Heladas medias (hasta -10ºC)
Platanus orientalis
Nombre común: Platano
Vegetación: Caducifolio
Altura: De 20 a 25 metros
Forma y estructura: Copa ovoidal. Tronco principal recto. Ramaje expandido
Color en primavera: Haz verde medio, enves verde claro
Color en otoño: Ocre
Resistencia a las heladas: Heladas fuertes (hasta -20ºC)
Quercus ilex
Nombre común: Encina
Vegetación: Perenne
Altura: En torno a 25 metros
Forma y estructura: Copa esférica o elíptica irregular. Tronco principal recto. Ramaje tortuoso
Color en primavera: Plateado en hojas jóvenes. En hojas antiguas, haz verde oscuro, envés plateado
Resistencia a las heladas: Heladas fuertes (hasta -15ºC)

6 Corrige y registra los errores garantizando que los documentos XML son válidos.

numeros.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE numeros [
  <!ELEMENT numeros (#PCDATA)>
]>
<numeros>
  <numero>25</numero>
</numeros>
```

letras.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE letras [
<!ELEMENT letras (letra)>
<!ELEMENT letra (#PCDATA)>
]>
<letras>
<letra>m</letra>
<letra>uve doble</letra>
</letras>
```

colores.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE colores [
<!ELEMENT colores (color*)>
<!ELEMENT color (#PCDATA)>
]>
<colores>
<color>azul marino</color>
negro
<color>amarillo</color>
/colores>
```

flores.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE flores [
<!ELEMENT flores (flor+)>
<!ELEMENT flor (#PCDATA)>
>
<flores>
</flores>
```

animales.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE animales [
  <!ELEMENT animales (animal*)>
  <!ELEMENT animal (#PCDATA)>
]>
<animales>
<perro>Caniche</perro>
<gato>Siamés</gato>
</animales>
```

escritores.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE escritores [
  <!ELEMENT escritores (escritor*)>
  <!ELEMENT escritor (nombre, nacimiento)>
  <!ELEMENT nombre (#PCDATA)>
  <!ELEMENT nacimiento (#PCDATA)>
]>
<escritores>
<escritor>
<nombre>Mario Vargas LLosa</nombre>
<nacimiento>28 de marzo de 1936</nacimiento>
</escritor>
<escritor>
<nacimiento>1 de abril de 1929</nacimiento>
<nombre>Milan Kundera</nombre>
</escritor>
</escritores>
```

musicos.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE musicos [
  <!ELEMENT musicos (musico*)>
  <!ELEMENT musico ((nombre | apodo), fechaNacimiento)>
  <!ELEMENT nombre (#PCDATA)>
  <!ELEMENT apodo (#PCDATA)>
  <!ELEMENT fechaNacimiento (#PCDATA)>
]>

<musicos>
<musico>
<nombre>Antonio Vivaldi</nombre>
<apodo>El cura pelirrojillo</apodo>
<fechaNacimiento>4 de marzo de 1678</fechaNacimiento>
</musico>
<musico>
<nombre>Johann Sebastian Bach</nombre>
<apodo>El viejo peluca</apodo>
<fechaNacimiento>21 de marzo de 1685</fechaNacimiento>
</musico>
</musicos>
```

telefonos_emergencia.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE agenda [
<!ELEMENT agenda (contacto*)>
<!ELEMENT contacto (nombre, telefonoFijo*, telefonoMovil+)>
<!ELEMENT nombre (#PCDATA)>
<!ELEMENT telefonoFijo (#PCDATA)>
<!ELEMENT telefonoMovil (#PCDATA)>
]>
<agenda>
<contacto>
<nombre>Ayuntamiento</nombre>
<telefonoFijo>010</telefonoFijo>
</contacto>
<contacto>
<nombre>Emergencias</nombre>
<telefonoFijo>112 (Unión Europea)</telefonoFijo>
<telefonoMovil>Desconocido</telefonoMovil>
<telefonoFijo>911 (Estados Unidos)</telefonoFijo>
</contacto>
</agenda>
```

sistema_solar.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE sistemaSolar [
  <!ELEMENT sistemaSolar (cuerpo*)>
  <!ELEMENT cuerpo (planeta|satelite|asteroide)>
  <!ELEMENT planeta (#PCDATA)>
  <!ELEMENT satelite (#PCDATA)>
  <!ELEMENT asteroide (#PCDATA)>
]>
<sistemaSolar>
  <cuerpo>
    <planeta>Tierra</planeta>
    <satelite>Luna</satelite>
  </cuerpo>
  <asteroide>Ceres</asteroide>
</sistemaSolar>
```

7 Modifica las descripciones DTD para conseguir documentos XML válidos.

marcadores.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE favoritos [
  <!ELEMENT favoritos (marcador)>
  <!ELEMENT marcador (nombre, uri)>
  <!ELEMENT nombre (#PCDATA)>
  <!ELEMENT uri (#PCDATA)>
]>

<marcadores>
<marcador>
<nombre>W3C</nombre>
<uri>http://www.w3.org/</uri>
</marcador>
<marcador>
<nombre>Web Hypertext Application Technology Working Group (WHATWG)</nombre>
<uri>http://www.whatwg.org/</uri>
</marcador>
</marcadores>
```

efemerides.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE efemerides [
  <!ELEMENT efemerides (efemeride)>
  <!ELEMENT efemeride (fecha, hecho)>
]>
<efemerides>
  <efemeride>
    <fecha>20 de julio de 1969</fecha>
    <hecho>Llegada del hombre a la Luna</hecho>
  </efemeride>
  <efemeride>
    <fecha>12 de octubre de 1492</fecha>
    <hecho>Llegada de Colón a América</hecho>
  </efemeride>
  <efemeride>
    <fecha>6 de abril de 1909</fecha>
    <hecho>llegada de Robert Peary al Polo Norte</hecho>
  </efemeride>
</efemerides>
```

aeropuertos.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE aeropuertos [
  <!ELEMENT aeropuertos (aeropuerto*)>
  <!ELEMENT aeropuerto (nombre, cerrado)>
  <!ELEMENT nombre (#PCDATA)>
  <!ELEMENT cerrado (#PCDATA)>
]>
<aeropuertos>
<aeropuerto>
<nombre>Berlín Schönefeld (SFX)</nombre>
</aeropuerto>
<aeropuerto>
<nombre>Berlín Tempelhof (THF)</nombre>
<cerrado />
</aeropuerto>
</aeropuertos>
```

vuelos.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE vuelos [
  <!ELEMENT vuelos (vuelo*)>
  <!ELEMENT vuelo (origen, destino)>
  <!ELEMENT origen (#PCDATA)>
  <!ELEMENT destino (#PCDATA)>
]>
<vuelos>
<vuelo>
<origen>Valencia (VLC)</origen>
<destino>Londres Heathrow (LHR)</destino>
</vuelo>
<vuelo>
<destino>Berlín Schönefeld (SFX)</destino>
<origen>Paris Charles de Gaulle (CDG)</origen>
</vuelo>
</vuelos>
```

reyes_esp.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE reyesEspañoles [
  <!ELEMENT reyesEspañoles (rey*, reina*)>
  <!ELEMENT rey (nombre, padre, madre)>
  <!ELEMENT nombre (#PCDATA)>
  <!ELEMENT padre (#PCDATA)>
  <!ELEMENT madre (#PCDATA)>
]>
<reyesEspañoles>
<rey>
<nombre>Felipe III</nombre>
<padre>Felipe II</padre>
<madre>Ana de Austria</madre>
</rey>
<reina>
<nombre>Juana la Loca</nombre>
<padre>Fernando el Católico</padre>
<madre>Isabel la Católica</madre>
</reina>
<rey>
<nombre>Carlos I</nombre>
<padre>Felipe el Hermoso</padre>
<madre>Juan la Loca</madre>
</rey>
</reyesEspañoles>
```

paises.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE paises [
  <!ELEMENT pais (nombre, unionEuropea, otan)>
  <!ELEMENT nombre EMPTY>
  <!ELEMENT unionEuropea EMPTY>
  <!ELEMENT otan EMPTY>
]>
<paises>
<pais>
<nombre>España</nombre>
<unionEuropea />
<otan />
</pais>
<pais>
<nombre>Noruega</nombre>
<otan />
</pais>
<pais>
<nombre>Austria</nombre>
<unionEuropea />
</pais>
</paises>
```

colores.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE colores [
  <!ELEMENT colores (color*)>
  <!ELEMENT nombreSvg (#PCDATA)>
  <!ELEMENT rgb (#PCDATA)>
  <!ELEMENT cmyk (#PCDATA)>
]>

<colores>
<color>
<nombreSvg>Purple</nombreSvg>
<codigo>
<rgb>#800080</rgb>
</codigo>
</color>
<color>
<nombreSvg>Purple</nombreSvg>
<codigo>
<cmyk>#00FF007F</cmyk>
</codigo>
</color>
</colores>
```

contabilidad.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE contabilidad [
  <!ELEMENT contabilidad ((ingreso | gasto)*)>
  <!ELEMENT fecha (#PCDATA)>
  <!ELEMENT cantidad (#PCDATA)>
  <!ELEMENT concepto (#PCDATA)>
]>
<contabilidad>
<apunte>
<ingreso />
<fecha>24 de febrero de 2011</fecha>
<cantidad>1800,00 €</cantidad>
<concepto>Salario</concepto>
</apunte>
<apunte>
<gasto />
<fecha>28 de febrero de 2011</fecha>
<cantidad>74,25 €</cantidad>
<concepto>Recibo luz</concepto>
</apunte>
</contabilidad>
```

mensajes.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE mensajes [
  <!ELEMENT mensajes (mensaje)>
  <!ELEMENT de (#PCDATA)>
  <!ELEMENT para (#PCDATA)>
  <!ELEMENT hora (#PCDATA)>
  <!ELEMENT texto (#PCDATA)>
  <!ELEMENT strong (#PCDATA)>
]>
<mensajes>
<mensaje>
<de>Pepe (pepe@example.com)</de>
<para>Juan (juan@example.com)</para>
<hora>28/02/2011 17:48:23,61</hora>
<texto>¿Hola, Juan, qué haces?</texto>
</mensaje>
<mensaje>
<de>Juan (juan@example.com)</de>
<para>Pepe (pepe@example.com)</para>
<hora>28/02/2011 17:54:20,87</hora>
 <texto>Aquí, aprendiendo <strong>XML</strong></texto>
</mensaje>
</mensajes>
```

8 Escribe la DTD que permita validar el documento XML que se muestra a continuación. Hacer dos versiones en cada caso: DTD externa e interna.

documento1.xml

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE nota SYSTEM "nota.dtd">
<nota>
<para>Pedro</para>
<de>Laura</de>
<titulo>Recordatorio</titulo>
<contenido>A las 7:00 en la puerta del teatro</contenido>
</nota>
```

documento2.xml

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<etiqueta>
<nombre>Roberto García</nombre>
<calle>c/ Mayor, 27</calle>
<ciudad>Coslada</ciudad>
<pais>España</pais>
<codigo>39343</codigo>
</etiqueta>
```

documento3.xml

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE lista_personas SYSTEM "lista_personas.dtd">
<lista_personas>
<persona>
<nombre>Antonio Moreno</nombre>
<fecha_nacimiento>27-11-2008</fecha_nacimiento>
<sexo>Varón</sexo>
</persona>
</lista_personas>
```

9 Escribe la DTD que permita validar el documento XML que se muestra a continuación. Haz dos versiones en cada caso: DTD externa e interna. Además, se sabe que siempre tiene que existir al menos un domicilio, y que el atributo "tipo" es obligatorio, y sólo puede tomar los valores "familiar" o "habitual".

documento4.xml

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE matricula SYSTEM "matricula.dtd">
<matricula>
<personal>
<dni>99223366M</dni>
<nombre>Juan Pardo Martín</nombre>
<titulacion>Ingeniería Informática</titulacion>
<curso_academico>1997/1998</curso_academico>
<domicilios>
<domicilio tipo="familiar">
<nombre>C/ Principal nº1</nombre>
</domicilio>
<domicilio tipo="habitual">
<nombre>C/ Secundaria nº2</nombre>
</domicilio>
</domicilios>
</personal>
<pago>
<tipo_matricula>Matrícula Ordinaria</tipo_matricula>
</pago>
</matricula>
```

10 Los siguientes documentos no son válidos porque contienen uno o dos errores (los errores no están en la DTD interna). Corrige los errores y consigue documentos XML válidos.

datos_personales.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE persona [
  <!ELEMENT persona EMPTY>
  <!ATTLIST persona nombre CDATA #IMPLIED>
]>

<persona dni="03141592E" />
```

pelicula.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE pelicula [
  <!ELEMENT pelicula EMPTY>
  <!ATTLIST pelicula titulo CDATA #IMPLIED>
]>

<pelicula titulo="La diligencia" genero="oeste" />
```

cuadros.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE cuadros [
  <!ELEMENT cuadros (cuadro*)>
  <!ELEMENT cuadro EMPTY>
  <!ATTLIST cuadro titulo ID #REQUIRED>
  <!ATTLIST cuadro autor CDATA #REQUIRED>
]>

<cuadros>
  <cuadro titulo="Adán y Eva" autor="Alberto Durero" />
  <cuadro autor="Lucas Cranach, el viejo" titulo="Adán y Eva" />
</cuadros>
```

compra.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE listaCompra [
  <!ELEMENT listaCompra (item*)>
  <!ELEMENT item EMPTY>
  <!ATTLIST item nombre CDATA #REQUIRED>
  <!ATTLIST item cantidad CDATA #REQUIRED>
]>

<listaCompra>
  <leche cantidad="12 litros" ></leche>
  <pan cantidad="3 barras de cuarto" />
</listaCompra>
```

jugadores_futbol.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE futbol [
  <!ELEMENT futbol (jugador*)>
  <!ELEMENT jugador EMPTY>
  <!ATTLIST jugador nombre NMTOKENS #REQUIRED>
  <!ATTLIST jugador codigo ID #REQUIRED>
]>

<futbol>
  <jugador nombre="Alfredo Di Stéfano" codigo="1"/>
  <jugador nombre="Edson Arantes do Nascimento, Pelé" codigo="2" />
  <jugador nombre="Diego Armando Maradona" codigo="3" />
  <jugador nombre="Johan Cruyff" codigo="4" />
</futbol>
```

jugadores_equipos.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE futbol [
  <!ELEMENT futbol ((jugador|equipo)*)>
  <!ELEMENT jugador EMPTY>
  <!ATTLIST jugador nombre NMTOKENS #REQUIRED>
  <!ATTLIST jugador codigo ID #REQUIRED>
  <!ELEMENT equipo EMPTY>
  <!ATTLIST equipo nombre CDATA #REQUIRED>
  <!ATTLIST equipo jugadores IDREFS #IMPLIED>
]>

<futbol>
  <jugador nombre="Alfredo Di Stéfano" codigo="ads"/>
  <jugador nombre="Edison Arantes do Nascimento" codigo="ean" />
  <jugador nombre="Diego Armando Maradona" codigo="dam" />
  <jugador nombre="Johan Cruyff" codigo="jc" />
  <equipo nombre="Società Sportiva Calcio Napoli" jugadores="Maradona" />
  <equipo nombre="Futbol Club Barcelona" jugadores="Cruyff, Maradona" />
</futbol>
```

libro.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE libro [
  <!ELEMENT libro EMPTY>
  <!ATTLIST libro autor NMTOKEN #REQUIRED>
]>

<libro autor="Mario Vargas LLosa" />
```

inventores.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE inventores [
  <!ELEMENT inventores>
  <!ELEMENT inventor EMPTY>
  <!ATTLIST inventor invento CDATA #REQUIRED>
  <!ATTLIST inventor nombre ID #REQUIRED>
]>

<inventores>
  <inventor nombre="Robert Adler" invento="Mando a distancia" />
  <inventor nombre="Laszlo Josef Biro" invento="Bolígrafo" />
  <inventor nombre="Josephine Garis Cochran" invento="Lavaplatos" />
  <inventor invento="Fuego" />
</inventores>
```

tareas.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE cosasPorHacer [
  <!ELEMENT cosasPorHacer (cosa)>
  <!ELEMENT cosa EMPTY>
  <!ATTLIST cosa fecha CDATA #REQUIRED>
  <!ATTLIST cosa asunto CDATA #REQUIRED>
  <!ATTLIST cosa fechaLimite CDATA #REQUIRED>
]>

<cosasPorHacer>
  <cosa fecha="20 de febrero de 2011" fechaLimite="1 de marzo de 2011">
  Preparar ejercicios de DTDs</cosa>
  <cosa fecha="21 de febrero de 2011" fechaLimite="5 de marzo de 2011">
  Preparar tema XSLT</cosa>
</cosasPorHacer>
```

resoluciones_pantalla.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE resoluciones [
  <!ELEMENT resoluciones EMPTY>
  <!ATTLIST resoluciones nombre NMTOKEN #REQUIRED>
  <!ATTLIST resoluciones alto CDATA #REQUIRED>
  <!ATTLIST resoluciones ancho CDATA #REQUIRED>
]>

<resoluciones>
  <resolucion nombre="VGA" alto="480" ancho="640" />
  <resolucion nombre="XGA" alto="1024" ancho="768" />
  <resolucion nombre="HD 1080" alto="1920" ancho="1080" />
</resoluciones>
```

comic.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE albumesMortadelo [
  <!ELEMENT albumesMortadelo (album*)>
  <!ELEMENT album (nombre, fecha)>
  <!ATTLIST album nombre CDATA #REQUIRED>
  <!ATTLIST album fecha(1969,1970,1971,1972,1973,1974) #REQUIRED>
]>

<albumesMortadelo>
  <album nombre="El sulfato atómico" fecha="1969"/>
  <album nombre="La caja de diez cerrojos" fecha="1971"/>
  <album nombre="El otro yo del profesor Bacterio" fecha="1973"/>
  <album nombre="Los cacharros majaretas" fecha="1974"/>
</albumesMortadelo>
```

11 Se quiere definir un lenguaje de marcas para representar los resultados de una liga de fútbol. La información que se quiere almacenar de cada partido es:

- El nombre del equipo local
- El nombre del equipo visitante
- Los goles marcados por el equipo local
- Los goles marcados por el equipo visitante

Escribe tres documentos que incluyan los siguientes resultados:

- Nottingham Presa: 0 - Inter de Mitente: 1
- Vodka Juniors: 3 - Sparta da Risa: 3
- Water de Munich: 4 - Esteaua es del grifo: 2

Cada documento incluirá un DTD diferente para representar ese lenguaje de marcas:

- DTD en la que no haya atributos, sino únicamente etiquetas
- DTD en la que los goles sean atributos
- DTD en la que toda la información se guarde en forma de atributos

XSD. XML Schema Definition.
Es una recomendación W3C.
El propósito de un esquema XML es definir la estructura y organización de un documento XML.
Es una alternativa más completa a las DTD para la validación de documentos XML.
Un documento **XSD es un documento XML.**
Muchos de los estándares XML actuales están definidos utilizando esquemas XML.

Ejemplo:

XML

DTD

```
<!ELEMENT note (to, from, heading, body)>
<!ELEMENT to (#PCDATA)>
<!ELEMENT from (#PCDATA)>
<!ELEMENT heading (#PCDATA)>
<!ELEMENT body (#PCDATA)>
```

```
<?xml version="1.0"?>
<note>
  <to>Tove</to>
  <from>Jani</from>
  <heading>Reminder</heading>
  <body>Don't forget me this weekend!</body>
</note>
```

XSD

```
<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">

<xs:element name="note">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="to" type="xs:string"/>
      <xs:element name="from" type="xs:string"/>
      <xs:element name="heading" type="xs:string"/>
      <xs:element name="body" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>

</xs:schema>
```

Espacio de nombres.

Según la W3C, un espacio de nombres XML es una colección de elementos y atributos XML identificados por un Identificador de recursos internacionalizado (IRI); esta colección a menudo se conoce como un "vocabulario" XML.

Una de las principales motivaciones para definir un espacio de nombres XML es evitar conflictos de nombres al usar y reutilizar vocabularios múltiples.
Los espacios de nombres son similares a los paquetes en Java.

Los espacios de nombres, se declaran como **atributo de un elemento**. Es posible declarar un espacio de nombres en cualquier elemento del documento y su alcance comienza en el elemento donde se declara y se aplica a todo el contenido de ese elemento, a menos que lo anule otra declaración de espacio de nombres.

```
<someElement xmlns: pfx = "http://www.foo.com" />
```

En el atributo ***xmlns: pfx***, ***xmlns*** es una palabra reservada utilizada para declarar el espacio de nombres y no queda vinculada a ningún espacio de nombres. Por lo tanto, el ejemplo anterior se lee vinculando el prefijo ***pfx*** con el espacio de nombres **"http://www.foo.com".** Siempre que se haga referencia al prefijo "***pfx***" dentro del alcance de la declaración de espacio de nombres, se expandirá al espacio de nombres real (***http://www.foo.com***) al que estaba vinculado:

Es una convención utilizar ***xsd*** o ***xs*** como prefijo para el espacio de nombres del esquema XML, pero esa decisión es puramente personal. Se puede elegir usar un prefijo ***ABC*** para el espacio de nombres del esquema XML, que es legal, pero no tiene mucho sentido.

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

Aunque un espacio de nombres generalmente se ve como una URL, eso no significa que uno deba estar conectado a Internet para declarar y usar espacios de nombres. Más bien, el espacio de nombres está destinado a servir como un "contenedor" virtual para el vocabulario y el contenido no mostrado que se puede compartir en el espacio de Internet. En el espacio de Internet, las URL son únicas, por lo tanto, usualmente elegirías usarlas para identificar espacios de nombres de manera única. Escribir la URL del espacio de nombres en un navegador no significa que muestre todos los elementos y atributos en ese espacio de nombres; es solo un concepto.

Los espacios de nombres W3C en la Recomendación XML imponen algunas restricciones de espacio de nombres:

- Los prefijos que comienzan con la secuencia de tres letras x, m y l, en cualquier combinación de caso, están reservados para su uso por XML y especificaciones relacionadas con XML. Aunque no es un error fatal, no es aconsejable vincular dichos prefijos. El prefijo xml está, por definición, vinculado al nombre del espacio de nombres:
  http://www.w3.org/XML/1998/namespace

- No se puede usar un prefijo a menos que esté declarado y vinculado a un espacio de nombres

Referencia

Para validar un documento XML con un esquema es necesario:

- Que el documento utilice una referencia al espacio de nombres (utilizando habitualmente el prefijo ***xsi***) siguiente:
  http://www.w3.org/2001/XMLSchema-instance.

- Referenciar en el documento XML el fichero de esquema que se va a utilizar para validación. Cuando el esquema no tiene asociado ningún espacio de nombres se utiliza el atributo ***nonameSchemaLocation***. El atributo ***schemaLocation*** asocia un documento de esquema que tiene un espacio de nombres de destino (***targetNamespace***) con un documento de instancia. Este atributo tendrá dos valores, separados por un espacio en blanco. El primer valor coincide con el del "**targetNamespace**" especificado en el schema. El segundo es la ubicación donde se encuentra definido el XML schema.

Ejemplos

```
<persona
    xmlns:p="http://www.prueba.es/persona"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:nonameSchemaLocation= "people.xsd">
    <nombre>Mercedes</nombre>
    <direccion>Arcos de la frontera</direccion>
</persona>
```

```xml
<p:persona
    xmlns:p="http://www.prueba.es/persona"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation= "http://www.prueba.es/persona people.xsd">
    <p:nombre>Mercedes</p:nombre>
    <p:direccion>Arcos de la frontera</p:direccion>
</p:persona>
```

Estructura XSD

```xml
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified">
  <xs:element name="root">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="element" type="xs:string" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

El nodo raíz es el elemento ***schema.***

targeNamespace. Atributo utilizado para hacer que nuestro esquema tenga un espacio de nombres propio.

elementFormDefault, ***attributeFormDefault***. Son atributos utilizados para especificar si los elementos y los atributos declarados en el esquema deben estar certificados por un espacio de nombres, ya sea explícitamente mediante un prefijo o implícitamente de forma predeterminada, cuando se utilizan en un documento instancia XML. Los posibles valores son:

- ***"qualified***": indica que, en los documentos instancia XML que referencien a este esquema, los elementos/atributos deben estar cualificados con un prefijo.
- "***unqualified":*** Este es el valor por defecto. Indica que los elementos/atributos no necesitan estar prefijados en el documento instancia.

Actividad guiada 1.

- Crea el siguiente fichero XML

```
<p:persona
    xmlns:p="http://www.midominio.es/persona"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation= "http://www.midominio.es/persona ejemplo4.xsd"
    >
    <p:nombre>John</p:nombre>
    <p:direccion>John</p:direccion>
</p:persona>
```

- Crea el siguiente fichero XSD

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://www.midominio.es/persona"
elementFormDefault="qualified"
attributeFormDefault="qualified">
  <xs:element name="persona">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="nombre" type="xs:string" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="direccion" type="xs:string" minOccurs="0"
            maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

- Comprueba que el documento está bien formado y es válido.
- Modifica el fichero XML para que esté bien formado y no sea válido.

Elementos

Los elementos son el bloque de construcción principal de cualquier documento XML.

Elementos simples.

Son elementos que **no pueden contener ningún otro elemento ni atributo**. Contienen **texto** de algunos de los tipos de datos **predefinidos** o **definidos** por el propio **usuario**.

Dentro del esquema (XSD) se definen de la siguiente manera:

```
<xs:element name="nombre" type="tipo"/>
```

name. Nombre del elemento en el documento XML.
type. Definición del tipo de elemento.

- ***xs:string***
- ***xs:integer***
- ***xs:boolean***
- ***xs:date***

El *tipo* puede ser definido por el usuario.

Detalle de tipos de datos

Ejemplo

```
<nombre>Mercedes</nombre>
<ciudad>Córdoba</ciudad>
<fecha_nacimiento>1970-03-27</fecha_nacimiento>

<xs:element name="nombre" type="xs:string"/>
<xs:element name="ciudad" type="xs:string"/>
<xs:element name="fecha_nacimiento" type="xs:date"/>
```

Actividad.

- Utilizando el ejemplo anterior crea un documento XML bien formado y válido.

El valor de un elemento simple puede verse afectado utilizando los siguientes atributos:
default. Significa que, si no se especifica ningún valor en el documento XML, la aplicación que lee el documento (generalmente un analizador XML o una biblioteca de enlace de datos XML) debe usar el valor predeterminado especificado en el XSD.
fixed. Significa que el valor en el documento XML solo puede tener el valor especificado en el XSD.

```
<xs:element name="ciudad" type="xs:string" default="Córdoba"/>
<xs:element name="forma_pago" type="xs:string" fixed="PayPal"/>
```

Atributos

Son declarados como tipos simples

```
<xs:attribute name="nombre" type="tipo"/>
```

name. Nombre del atributo en el documento XML
type. Descripción del elemento. Hay varios tipos predefinidos

xs:string
xs:decimal
xs:integer
xs:boolean
xs:date
xs:time

Ejemplo

```
<xs:attribute name="lang" type="xs:string"/>
<titulo lang="ES">Cien Años de Soledad</titulo>
```

El valor del atributo puede verse afectado utilizando los siguientes atributos.
default. Se utiliza para establecer un valor por defecto en el atributo. Si no se especifica ningún valor en el documento XML, la aplicación que lee el documento (generalmente un analizador XML o una biblioteca de enlace de datos XML) debe usar el valor especificado en el XSD.
fixed. Se utiliza para fijar el valor del atributo, no pudiendo ser distinto al especificado en el XSD.
use: Se utiliza para establecer la obligatoriedad o no del atributo. Por defecto es opcional. Valores permitidos son:

- required
- optional

```
<xs:attribute name="lang" type="xs:string" default="ES"/>
<xs:attribute name="lang" type="xs:string" fixed="ES"/>
<xs:attribute name="lang" type="xs:string" use="required"/>
```

Restricciones.

XML Schema permite definir restricciones a los posibles valores de los tipos de datos. Dichas restricciones se pueden establecer en diferentes aspectos, llamados facetas.

```
<xs:element name="nota">
 <xs:simpleType>
  <xs:restriction base="xs:integer">
   <xs:minInclusive value="1"/>
   <xs:maxInclusive value="9"/>
  </xs:restriction>
 </xs:simpleType>
</xs:element>
```

Las facetas permiten definir restricciones sobre los posibles valores de atributos o elementos. Las facetas que pueden utilizarse son:
xs:*enumeration*. Especifica una lista de valores admitidos.
xs:fractionDigits. Especifica el número máximo de decimales que puede tener un número.
Xs:totalDigits. Especifica el número máximo de dígitos que puede tener un número.
xs:length: Especifica una longitud fija.
xs:minLength: Especifica una longitud mínima.
xs:maxLength: Especifica una longitud máxima.
xs:maxExlucisve. Especifica que el valor debe ser menor que el indicado.
xs:maxInclusive. Especifica que el valor debe ser menor o igual que el indicado.
xs:minExclusive. Especifica que el valor debe ser menor que el indicado.
xs:minInclusive. Especifica que el valor debe ser mayor o igual que el indicado.
xs:pattern: Especifica un patrón de caracteres admitidos.
xs:whiteSpace. Especifica cómo se debe tratar a los posibles espacios en blanco, tabulaciones, saltos de línea y los retornos de carro que puedan aparecer.

Ejemplo:

```
<xs:element name="forma_pago">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:enumeration value="Contado"/>
      <xs:enumeration value="Tarjeta"/>
      <xs:enumeration value="PayPal"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
```

El mismo ejemplo definiendo un elemento tipo:

```
<xs:element name="forma_pago" type="formaPagoType"/>

<xs:simpleType name="formaPagoType">
  <xs:restriction base="xs:string">
    <xs:enumeration value="Contado"/>
    <xs:enumeration value="Tarjeta"/>
    <xs:enumeration value="PayPal"/>
  </xs:restriction>
</xs:simpleType>
```

Elementos complejos.

Elementos XML que contienen otros elementos y/o atributos. Hay cuatro tipos de elementos complejos que pueden contener también atributos.

- Elementos vacíos.
- Elementos que contienen solo otros elementos.
- Elementos con atributos que contienen solo texto.
- Elementos que contienen tanto otros elementos como texto.

➢ Escribir un ejemplo de cada uno de ellos.

Hay dos formas de declarar un elemento complejo.

1 En la declaración del elemento.

```
<xs:element name="empleado">
 <xs:complexType>
  <xs:sequence>
   <xs:element name="nombre" type="xs:string"/>
   <xs:element name="apellido" type="xs:string"/>
  </xs:sequence>
 </xs:complexType>
</xs:element>
```

2 Declarando el tipo y utilizándolo como valor del atributo ***type***.

```
<xs:complexType name="personaTipo">
 <xs:sequence>
  <xs:element name="nombre" type="xs:string"/>
  <xs:element name="apellidos" type="xs:string"/>
 </xs:sequence>
</xs:complexType>
```

En este caso puede utilizarse para definir varios elementos de este tipo.

```
<xs:element name="alumno" type="personaTipo"/>
<xs:element name="profesor" type="personaTipo"/>
```

Ejemplos

Elemento complejo vacío.

```
<producto codigo="1234" />
```

Declaración.

```
<xs:element name="producto">
 <xs:complexType>
  <xs:attribute name="codigo" type="xs:positiveInteger"/>
 </xs:complexType>
</xs:element>
```

Elemento complejo con solo elementos.

```
<persona>
 <nombre>John</nombre>
 <apellidos>Smith</apellidos>
</persona>
```

Declaración

```
<xs:element name="persona">
 <xs:complexType>
  <xs:sequence>
   <xs:element name="nombre" type="xs:string"/>
   <xs:element name="apellidos" type="xs:string"/>
  </xs:sequence>
 </xs:complexType>
</xs:element>
```

- Declara de nuevo el elemento definiendo un nuevo tipo.

Elemento complejo con contenido mixto.

```
<letter>
 Dear Mr. <name>John Smith</name>.
 Your order <orderid>1032</orderid>
 will be shipped on <shipdate>2001-07-13</shipdate>
</letter>

<xs:element name="letter">
 <xs:complexType mixed="true">
  <xs:sequence>
   <xs:element name="name" type="xs:string"/>
   <xs:element name="orderid" type="xs:positiveInteger"/>
   <xs:element name="shipdate" type="xs:date"/>
  </xs:sequence>
 </xs:complexType>
</xs:element>
```

- Declarar utilizando un nuevo tipo.

```
<xs:element name="letter" type="lettertype"/>

<xs:complexType name="lettertype" mixed="true">
 <xs:sequence>
  <xs:element name="name" type="xs:string"/>
  <xs:element name="orderid" type="xs:positiveInteger"/>
  <xs:element name="shipdate" type="xs:date"/>
 </xs:sequence>
</xs:complexType>
```

Extensiones

Permiten definir nuevos elementos utilizando como base un elemento simple o complejo.

xs:extension. Se utiliza para extender un elemento ***simpleType*** o ***complexType***.
xs:simpleContent. Se utiliza para definir restricciones o extensiones a elementos simples, es decir elementos que solo contienen datos.
xs:complexContent. Se utiliza para definir restricciones o extensiones a un tipo complejo.

Ejemplos:

Definición de un elemento complejo como base para definir nuevos elementos complejos.

```
<xs:element name="alumno" type="alumnoTipo"/>

<xs:complexType name="personaTipo">
 <xs:sequence>
  <xs:element name="nombre" type="xs:string"/>
  <xs:element name="apellidos" type="xs:string"/>
 </xs:sequence>
</xs:complexType>

<xs:complexType name="alumnoTipo">
 <xs:complexContent>
  <xs:extension base="personaTipo">
   <xs:sequence>
    <xs:element name="nie" type="xs:string"/>
    <xs:element name="curso" type="xs:string"/>
   </xs:sequence>
  </xs:extension>
 </xs:complexContent>
</xs:complexType>
```

Definición de un elemento complejo que sólo incluye texto. Es necesario agregar

un elemento ***simpleContent*** para agrupar el contenido y que incluya una extensión o una restricción.

```
<shoesize country="france">35</shoesize>

<xs:element name="shoesize">
 <xs:complexType>
  <xs:simpleContent>
   <xs:extension base="xs:integer">
    <xs:attribute name="country" type="xs:string" />
   </xs:extension>
  </xs:simpleContent>
 </xs:complexType>
</xs:element>
```

La definición del elemento anterior utilizando el atributo ***type***.

```
<xs:element name="shoesize" type="shoetype"/>

<xs:complexType name="shoetype">
 <xs:simpleContent>
  <xs:extension base="xs:integer">
   <xs:attribute name="country" type="xs:string" />
  </xs:extension>
 </xs:simpleContent>
</xs:complexType>
```

Indicadores

Se utilizan para establecer cómo se van a utilizar los elementos en un documento XML. Hay siete tipos de indicadores que se pueden clasificar en:

- Indicadores de orden: secuencia (sequence), todo (all) y elección (choice).
  xs:sequence: Especifica el orden en el que obligatoriamente deben aparecer los elementos hijo de un elemento.
  xs:all: Sirve para indicar que dichos elementos pueden aparecer en cualquier orden
  xs:choice: Permite especificar que solamente se permite escribir uno de los elementos hijo.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
 <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>
</xs:schema>
```

- Indicadores de ocurrencia: ***maxOccurs*** y ***minOccurs***.
  Se utilizan para establecer, respectivamente, el número máximo y mínimo de veces que puede aparecer un determinado elemento. El valor por defecto es 1. A ***minOccurs*** se le puede asignar cualquier valor entero no negativo y a ***maxOccurs*** se le puede asignar cualquier valor entero no negativo o la constante de cadena "***unbounded***".

```
<xs:element name="Customer_order"
            type="xs:integer"
            minOccurs ="0"
            maxOccurs="unbounded"/>
```

```
<xs:element name="redes_sociales"
            type="xs:string"
            minOccurs="1"
            maxOccurs="5"/>
```

- Indicadores de grupo: Se utilizan para agrupar un conjunto de declaraciones de

elementos (***group***) y de atributos (***attributeGroup***)
xs:group. Se utiliza para definir una agrupación de conjunto de declaraciones de elementos relacionados.
xs:attributeGroup. Se utiliza para definir una agrupación de atributos.

```
<xs:attributeGroup name="personattr">
  <xs:attribute name="attr1" type="string"/>
  <xs:attribute name="attr2" type="integer"/>
</xs:attributeGroup>

<xs:complexType name="person">
  <xs:attributeGroup ref="personattr"/>
</xs:complexType>
```

Actividad.

Dado el siguiente fichero XML y considerando que se quiere especificar que:

"país" pueda aparecer una o ilimitadas veces.

"nombre" tenga que escribirse obligatoriamente, y solo una vez, dentro de país".

De cada "país" puedan escribirse de cero a cinco "ciudades".

```
<?xml version="1.0" encoding="UTF-8"?>
<paises xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:noNamespaceSchemaLocation="paises.xsd">
  <pais>
    <nombre>Argentina</nombre>
    <ciudad>Buenos Aires</ciudad>
    <ciudad>Rosario</ciudad>
  </pais>
  <pais>
    <nombre>México</nombre>
    <ciudad>Guadalajara</ciudad>
    <ciudad>Monterrey</ciudad>
    <ciudad>Cancún</ciudad>
    <ciudad>Mérida</ciudad>
    <ciudad>Ciudad de México</ciudad>
  </pais>
  <pais>
    <nombre>Colombia</nombre>
  </pais>
</paises>
```

- Escribir el XSD para validar

Solución.

Actividad.
Dado el siguiente fichero XSD, escribir un fichero XML válido.

```
<?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>
```

Solución.

Actividad.
Escribir un fichero XSD para validar el siguiente documento XML realizando un agrupamiento de atributos.

```
<?xml version="1.0" encoding="UTF-8"?>
<personas xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="personas.xsd">
  <persona nombre="Eva" edad="25" pais="Francia"/>
  <persona nombre="Giovanni" edad="26" pais="Italia"/>
</personas>
```

Solución

Actividad

```
<?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" maxOccurs="unbounded">
          <xs:complexType mixed="true">
            <xs:sequence>
              <xs:element name="nombre" type="xs:string"/>
              <xs:element name="ciudad" type="xs:string"/>
              <xs:element name="edad" type="xs:positiveInteger"/>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Utilizando los elementos "nombre", "ciudad" y "edad", escribir el código de un documento XML que pueda ser validado por "personas.xsd" y que almacene la siguiente información:

"Eva vive en París y tiene 25 años."
"Giovanni vive en Florencia y tiene 26 años."

Solución

Actividades

1 Escribe las definiciones de los siguientes elementos <ejr1>

```
<ciudad>Roma</ciudad>
<fecha-de-nacimiento>1996-12-18</fecha-de-nacimiento>
<hora>18:29:45</hora>
<nota>7.5</nota>
<apto>true</apto>
```

2 Definir un elemento llamado puertaCerrada de tipo lógico, que por defecto tenga el valor "falso", y otro elemento llamado ventanaAbierta también de tipo lógico, que tenga asignado el valor fijo "verdadero". <

3 Escribir un fichero XSD para validar el siguiente documento XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<fichas>
  <ficha numero="1">
    <nombre>Ana Sanz Tin</nombre>
    <edad>22</edad>
  </ficha>
  <ficha numero="2">
    <nombre>Iker Rubio Mol</nombre>
    <edad>23</edad>
  </ficha>
</fichas>
```

4 Se desea almacenar información sobre personas de las que almacenamos en este orden:

datos: Incluyen nombre, apellidos y dni.
comentario: Breve descripción de la persona.
Toda persona tiene necesariamente una fecha de nacimiento como atributo.

Escribir un fichero XML y un esquema de validación.

5 Se desea almacenar información sobre personas de las que almacenamos en este orden:

datos: Incluyen nombre, apellidos y dni.
comentario: Breve descripción de la persona.
Toda persona tiene opcionalmente una fecha de nacimiento como atributo.
Almacenamos también información sobre forma de pago, que puede ser con tarjeta y almacenamos el número de tarjeta o con transferencia y almacenamos el número de cuenta bancaria

Escribir un fichero XML y un esquema de validación.

6 Modificar el esquema de validación anterior para permitir cualquier orden en los elementos.

7 Modificar el documento XML y esquema de validación anterior para incluir como información una nota con valores comprendidos entre 0 y 10.

8 Dado el siguiente fichero XML

```
<?xml version="1.0" encoding="UTF-8"?>
<examenes xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                xsi:noNamespaceSchemaLocation="ejr3.xsd">
  <examen numero="1">
    <nombre>Raúl García</nombre>
    <nota>6</nota>
  </examen>
  <examen numero="2">
    <nombre>Miguel Feria</nombre>
    <nota>8</nota>
  </examen>
</examenes>
```

Escribir un esquema para validarlo, teniendo en cuenta que la nota es un valor entre 1 y 10

9 Dado el siguiente documento XML

```
<?xml version="1.0" encoding="UTF-8"?>
<precios xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="tarifas.xsd">
  <temporada_alta>170.25</temporada_alta>
  <temporada_baja>111.23</temporada_baja>
  <temporada_media>100.00</temporada_media>
</precios>
```

Escribir un esquema para validarlo, teniendo en cuenta que los precios pueden tomar como máximo un número de 5 dígitos, de los cuales solo dos pueden ser decimales.

10 Dado el siguiente fichero XSD

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
 <xs:element name="fichas">
  <xs:complexType>
   <xs:sequence>
    <xs:element name="ficha" maxOccurs="unbounded">
     <xs:complexType>
      <xs:sequence>
       <xs:element name="nombre" type="xs:string"/>
       <xs:element name="iniciales">
        <xs:simpleType>
         <xs:restriction base="xs:string">
          <xs:pattern value="[A-Z][A-Z][A-Z]"/>
         </xs:restriction>
        </xs:simpleType>
       </xs:element>
       <xs:element name="edad" type="xs:integer"/>
      </xs:sequence>
     </xs:complexType>
    </xs:element>
   </xs:sequence>
  </xs:complexType>
 </xs:element>
</xs:schema>
```

Corrige el siguiente documento XML para que sea válido.

```
<?xml version="1.0" encoding="UTF-8"?>
<fichas xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="fichas.xsd">
  <ficha>
    <nombre>Antonio Machado Ruiz</nombre>
    <iniciales>AMR</iniciales>
    <edad>22</edad>
  </ficha>
  <ficha>
    <nombre>Mario Moreno</nombre>
    <iniciales>MM</iniciales>
    <edad>23</edad>
  </ficha>
  <ficha>
    <iniciales>ALO</iniciales>
    <nombre>Ada Lovelace</nombre>
    <edad>24</edad>
  </ficha>
  <ficha>
    <nombre>pablo ruiz picasso</nombre>
    <iniciales>prp</iniciales>
    <edad>24</edad>
  </ficha>
</fichas>
```

EJERCICIOS profesor. Ejercicios que no están en el documento del alumno. Utilizados para prácticas en clase individuales y exámenes.

- Define un documento XSD para validar documentos XML que describan una cadena que represente un número binario. El elemento raíz contiene una serie de elementos uno y cero. El elemento uno debe contener 1 y el elemento cero debe contener 0

```
<numero_binario>
   <uno>1</uno>
   <uno>1</uno>
   <cero>0</cero>
</numero_binario>
```

Ámbitos de aplicación.

SVG Scalable Vector Graphics

Recursos

- Inskape
- Open Clip Art
- publicdomainvectors.org
- SVG en W·C

SVG es un formato de gráficos vectoriales bidimensionales, tanto estáticos como animados, en formato XML, cuya especificación es un estándar abierto desarrollado por el W3C desde el año 1999.

SVG se convirtió en una recomendación del W3C en septiembre de 2001.

SVG ha sido incorporado de forma nativa en la mayoría de los navegadores actuales, por lo que no es necesario ningún complemento para visualizarlos en ellos.

Ejemplo.

- Creamos el siguiente documento SVG

```
<svg version="1.1"
     baseProfile="full"
     width="300" height="200"
     xmlns="http://www.w3.org/2000/svg">
  <rect width="100%" height="100%" fill="red" />
  <circle cx="150" cy="100" r="80" fill="green" />
  <text x="150" y="125" font-size="60"
        text-anchor="middle" fill="white">
        SVG
  </text>
</svg>
```

- Abrimos el documento con el navegador.

En un gráfico vectorial, los elementos de la imagen están definidos como formas elementales (líneas, rectángulos, círculos, curvas, polígonos, etc.), definidas como elementos XML.

Los dibujos SVG se definen en un plano infinito en el que el eje Y está orientado hacia abajo.

Por ello SVG no es un formato adecuada para fotografías, pero es idóneo para cualquier tipo de dibujo, técnico o artístico.

Algunas ventajas.

- Las imágenes SVG se pueden ampliar a cualquier escala sin perder calidad.
- Las imágenes SVG suelen ocupar poco espacio, ya que están definidas mediante etiquetas. El tamaño en KB de la imagen es además independiente del tamaño con el que se ve en la página web.
- Las imágenes son fácilmente combinables y reutilizables.
- En una web las imágenes se pueden modificar de forma dinámica mediante hojas de estilo o Javascript ya que forman parte de las mismas.
- Un gráfico SVG puede incluirse fácilmente en un documento HTML como objeto interno o como objeto externo.

Algunos inconvenientes.

- La implementación en los navegadores no siempre se realiza de la forma más rápida.
- Riesgos a nivel de seguridad al permitir código Javascript en los documentos.

Elemento raíz.

<svg> Elemento raíz del documento, engloba por tanto toda la imagen y las formas que la componen.
Atributos

- ***versión***
- ***xmlns***
- ***viewBox***. Establece la porción del plano SVG que muestra la imagen.
- ***width***
- ***height***

Figuras simples.

- ***<line>***
- ***<rect>***
- ***<circle>***
- ***<ellipse>***
- ***<text>***
- ***<path>***

Estilos SVG

SVG permite trabajar con estilos utilizado algunos atributos de los elementos.

Existen dos formas de aplicar estilos a un elemento SVG: Dentro de una regla de estilo CSS o directamente con atributos del elemento.

- ***fill***. Se utiliza para aplicar color de relleno. Puede utilizarse una constante (red), un valor hexadecimal (#222222) o un valor RGB (rgb(19,133,198))
- **stroke**. Fija el color del trazo.
- ***stroke-width***. Tamaño de trazo, usualmente en píxeles.
- ***opacity***. Permite establecer la opacidad y permite un valor entre 0 y 1

Para el elemento ***text***, algunos atributos válidos son:

- ***font-family***
- ***font-size***

Ejemplo:

```
<circle
     cx="25" cy="25" r="22"
     fill="yellow" stroke="orange" stroke-width="5"/>
```

Utilizando CSS

```
.pumpkin {
    fill: yellow;
    stroke: orange;
    stroke-width: 5;
    }
. . .

<circle cx="25" cy="25" r="22" class="pumpkin"/>
```

Ejemplos:

Ejemplo 1. Crea el siguiente documento y ábrelo con el navegador.

```
<svg  xmlns="http://www.w3.org/2000/svg"
      xmlns:xlink="http://www.w3.org/1999/xlink">
    <rect x="10" y="10" height="100" width="100"
          style="stroke:#ff0000; fill: #0000ff"/>
</svg>
```

Ejemplo 2. Crea el siguiente documento y ábrelo con el navegador.

```
<svg xmlns="http://www.w3.org/2000/svg"
    xmlns:xlink="http://www.w3.org/1999/xlink">
    <line x1="0"  y1="10" x2="0"   y2="100" style="stroke:#006600;"/>
    <line x1="10" y1="10" x2="100" y2="100" style="stroke:#006600;"/>
    <line x1="20" y1="10" x2="100" y2="50"  style="stroke:#006600;"/>
    <line x1="30" y1="10" x2="110" y2="10"  style="stroke:#006600;"/>
</svg>
```

Ejemplo 3. Crea el siguiente documento y ábrelo con el navegador.

```
<svg xmlns="http://www.w3.org/2000/svg"
     xmlns:xlink="http://www.w3.org/1999/xlink">

     <rect x="10" y="10" height="110" width="110"
           style="stroke:#ff0000; fill: #0000ff">
          <animateTransform
               attributeName="transform"
               begin="0s"
               dur="20s"
               type="rotate"
               from="0 60 60"
               to="360 60 60"
               repeatCount="indefinite"
          />
     </rect>
</svg>
```

Ejemplo 4. Completa el siguiente documento.

```
<svg xmlns="http://www.w3.org/2000/svg"
    xmlns:xlink="http://www.w3.org/1999/xlink">

    <rect x="50" y="50" height="110" width="110"
          style="stroke:#ff0000; fill: #ccccff"
          transform=  /* Aplica transform*/
            >
    </rect>
    <text x="70" y="100"
          transform=/* Aplica transform*/
    >Hello World</text>
</svg>
```

Actividades.

1 Diseña una imagen de marca vectorial para tu sitio web.
2 Diseña y muestra en una página web las cuatro fichas del parchís.

KML, Keyhole Markup Language

Recursos.

Tutorial en Google Developers.

Lenguaje de marcado basado en XML para representación de información geográfica.

En 2008 la versión 2.2 es adoptada por el OGC (Open Geospatial Consortium) como estándar abierto de intercambio de información geográfica.

Estructura KML

```
<?xml version="1.0" encoding="UTF-8"?>
<kml xmlns="http://earth.google.com/kml/2.0"> <Document>
<Placemark>
 <Point>
  <coordinates>
     -4.7793869,37.8789056,0
  </coordinates>
 </Point>
</Placemark>
</Document>
</kml>
```

- Probamos el código.

Marcas de posición.

Las marcas de posición (Placemark) permiten marcar una posición en la superficie de la Tierra con un icono de chincheta amarilla. La marca de posición (Placemark) más sencilla incluye solo un elemento de punto (<Point>), que especifica la ubicación de la marca de posición. Puedes especificar un nombre y un icono personalizado para una marca de posición y, si quieres, le puedes añadir otros elementos geométricos.

```
<?xml version="1.0" encoding="UTF-8"?>
<kml xmlns="http://www.opengis.net/kml/2.2">
<Placemark>
 <name>Marca de posición simple</name>
 <description>Se coloca pegada al suelo.</description>
 <Point>
 <coordinates>-122.0822035425683,37.42228990140251,0</coordinates>
 </Point>
 </Placemark>
</kml>
```

Superposiciones del suelo

Las superposiciones de suelo permiten "colocar" una imagen sobre el relieve de la Tierra. El elemento de icono (<Icon>) incluye el enlace al archivo de imagen que contiene la imagen de superposición.

Ejemplo que muestra la erupción del volcán Etna en el año 2001:

```
<?xml version="1.0" encoding="UTF-8"?>
<kml xmlns="http://www.opengis.net/kml/2.2"> <Folder>
 <name>Superposiciones de suelo</name>
 <description>Ejemplos de superposiciones de suelo</description>
 <GroundOverlay>
 <name>Superposición a gran escala sobre relieve</name>
 <description>La superposición muestra la erupción del Etna el 13 de julio de 2001.</description>
 <Icon>
 <href>http://developers.google.com/kml/documentation/images/etna.jpg</href>
 </Icon>
 <LatLonBox>
 <north>37.91904192681665</north>
 <south>37.46543388598137</south>
 <east>15.35832653742206</east>
 <west>14.60128369746704</west>
 <rotation>-0.1556640799496235</rotation>
 </LatLonBox>
 </GroundOverlay>
 </Folder>
 </kml>
```

- Probamos el fichero.

Rutas

Se pueden crear muchos tipos de rutas en Google Earth y es fácil ser creativo con los datos. En KML, las rutas se crean con el elemento de cadena de líneas (<LineString>).

Ejemplo

```
<?xml version="1.0" encoding="UTF-8"?>
<kml xmlns="http://www.opengis.net/kml/2.2"> <Document>
 <name>Rutas</name>
 <description>Ejemplos de rutas. Observa que la etiqueta tessellate se establece de forma predeterminada
 en 0. Si quieres crear líneas teseladas, deben crearse
 (o editarse) directamente en KML.</description> <Style id="yellowLineGreenPoly">
 <LineStyle>
 <color>7f00ffff</color>
 <width>4</width>
 </LineStyle>
 <PolyStyle>
 <color>7f00ff00</color>
 </PolyStyle>
 </Style> <Placemark>
 <name>Relieve absoluto</name>
 <description>Pared verde transparente con contornos amarillos</description>
```

```
<styleUrl>#yellowLineGreenPoly</styleUrl>
<LineString>
<extrude>1</extrude>
<tessellate>1</tessellate>
<altitudeMode>absoluto</altitudeMode>
<coordinates> -112.2550785337791,36.07954952145647,2357
-112.2549277039738,36.08117083492122,2357
-112.2552505069063,36.08260761307279,2357
-112.2564540158376,36.08395660588506,2357
-112.2580238976449,36.08511401044813,2357
-112.2595218489022,36.08584355239394,2357
-112.2608216347552,36.08612634548589,2357
-112.262073428656,36.08626019085147,2357
-112.2633204928495,36.08621519860091,2357
-112.2644963846444,36.08627897945274,2357
-112.2656969554589,36.08649599090644,2357 </coordinates>
</LineString> </Placemark>
</Document> </kml>
```

- Probamos el fichero.

Actividades.

http://dagik.org/kml_intro/E/point.html
Partiendo del ejercicio inicial.
Colocar marcador en el insittuto
Añadir nombre al marcador. I.E.S. Gran Capitán.
Añadir una descripción al marcador.
Departamento de informática
Cambiar icono al marcador. Hay que añadir la imagen y comprimir los dos ficheros con extensión kmz
Cambiar el estilo de la ficha
Poner marca en altitud

Actividades.

Crear una ruta que turística por Córdoba.