

LENGUAJES DE MARCAS Y SISTEMAS DE GESTIÓN DE INFORMACIÓN



Elena Fernández Chirino

UNIDAD 4:

Definición de esquemas y vocabularios en XML

INDICACIONES TAREA 4 : RELACIÓN CON LA TEORÍA

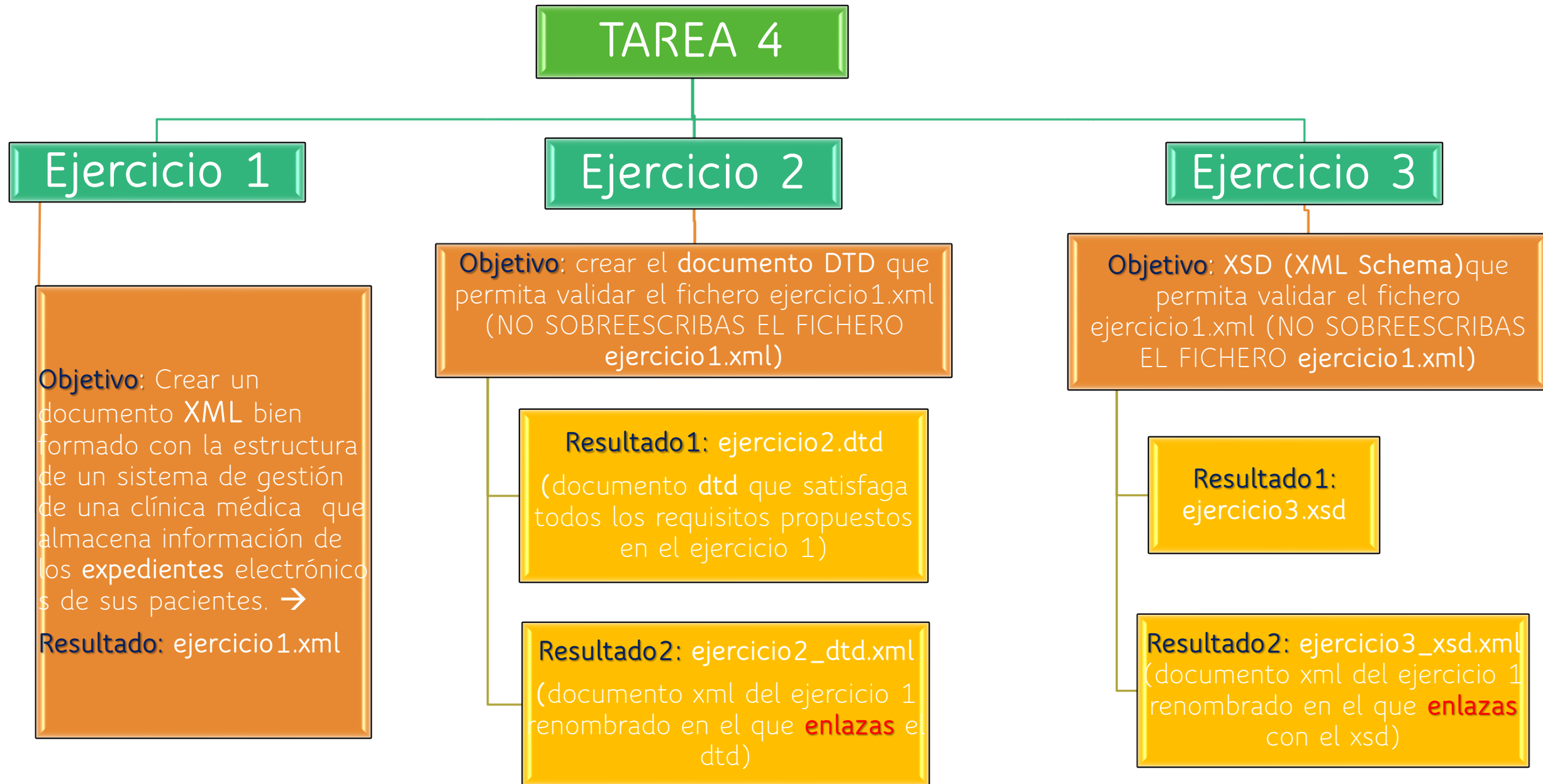
DEPARTAMENTO DE INFORMÁTICA

<http://www.iestrassierra.com>



This work is licensed under a [Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License](https://creativecommons.org/licenses/by-nc-sa/4.0/).

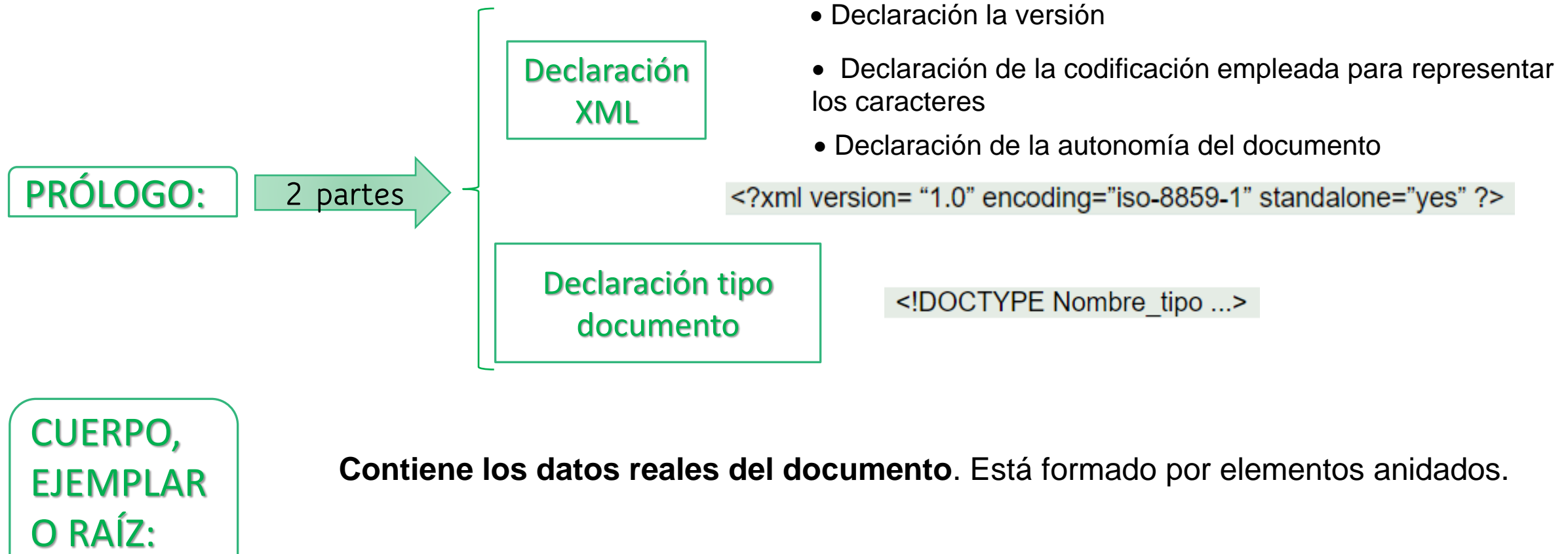
1.- INDICACIONES GENERALES TAREA 4



Ejercicio 1

¿Qué parte de la unidad se está trabajando es esta actividad?

Estructura de un documento XML bien construido (Unidad 1)



Ejercicio 1

Nuevo concepto: Espacio de nombres o NAMESPACES

ESPACIO DE NOMBRES

- Los **espacios de nombres en XML (XML namespaces)** se utilizan para evitar conflictos de nombres entre elementos y atributos en un documento XML. Permiten asignar un identificador único a cada elemento y atributo, incluso si tienen el mismo nombre, pero pertenecen a diferentes vocabularios o dominios.
- Para más información: [Espacios de nombres en XML - Eniun](#)
- **Por ejemplo**, varios documentos XML se pueden combinar entre sí, pudiendo en estos casos coincidir el nombre de algunos elementos. Dos documentos XML podrían contener un elemento llamado “carta”, pero con significados distintos.
- [Espacios de nombres en XML | Tutorial de XML | Abrirllave.com](#)

NOTA:

No es obligatorio usar namespaces en la Tarea 4, pero sí **tenemos que aprender el concepto en esta unidad.**

EJERCICIOS 2 Y 3

¿Todo documento XML **bien formado** es **válido**? NO

VALIDACIÓN DE UN DOCUMENTO XML

- Cuando un documento XML **cumple estrictamente las reglas generales de creación XML** se dice que **está bien formado**.
- Cuando además sigue las reglas de un documento de validación entonces se dice que **es válido**.

TÉCNICAS PARA VALIDAR UN DOCUMENTO XML:

Las técnicas más populares para validar documentos son:

- **DTD** (Document Type Definition) → Se deberá realizar en el ejercicio 2
- **XML** (Schema o esquemas XML) → Se deberá realizar en el ejercicio 3

EJERCICIO 2

Generar un documento DTD que valide el ejercicio 1

- ✓ 1º Crearemos el documento DTD: **ejercicio2.dtd**
- ✓ 2º Copiamos el ejercicio1.xml y lo renombraremos como: **ejercicio2_dtd.xml** y lo modificaremos para enlazarlo con **ejercicio2.dtd**
- ✓ Tipos de DTD:

DTD incrustada: Es posible incluir la DTD en el mismo documento XML, en concreto en el prólogo, después de la declaración XML.

DTD externa: Es posible separar ambos elementos, guardándolos en archivos diferentes. Así, se puede situar en un archivo aparte la DTD y después enlazarlo con el archivo XML mediante **<!DOCTYPE nombre SYSTEM "archivo.dtd">**



En el ejercicio 2 lo haremos de esta forma

EJERCICIO 3

Generar un XML Schema (Extensión del archivo .xsd)

- ✓ 1º Crearemos el documento XSD: **ejercicio3.xsd**
- ✓ 2º Copiamos el ejercicio1.xml y lo renombraremos como: **ejercicio3_xsd.xml** y lo modificaremos para enlazarlo con **ejercicio3.xsd**
- ✓ **Nota inicial:** Los elementos y atributos XML que se utilizan para generar esquemas pertenecen al espacio de nombres XML Schema, con el URI: <http://www.w3.org/2001/XMLSchema>.
- ✓ Un documento XML Schema tiene como **elemento raíz**, ejemplar, un elemento **<xs:schema>**. Este elemento contendrá las declaraciones de todos los elementos y atributos que puedan aparecer en un documento XML asociado. Este elemento tendrá un elemento hijo que será el elemento raíz del documento XML, que declararemos con **<xs:element>**.
- ✓ Por tanto, nuestro documento .xsd deberá comenzar:

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

EJERCICIO 2: DTD

Los DTDs describen la estructura de los documentos XML mediante declaraciones

Tipos de declaraciones DTD

- Declaraciones de **entidades**: permite **definir atajos para caracteres especiales o textos largos** que se utilizan frecuentemente.

Generales

Internas

Externas

De parámetro o paramétrica

Internas

Externas

- Declaraciones de **notaciones**. Cuando se incluyen ficheros binarios en un fichero, se indica a la aplicación cómo ha de hacerse cargo de ellos utilizando notaciones. La sintaxis para declarar notaciones es: `<!NOTATION nombre SYSTEM aplicacion>` → EJEMPLO:

- Declaraciones de **elementos**.

Terminales

No terminales

- Declaraciones de **atributos**.

LAS MÁS UTILIZADAS

EJERCICIO 2: DTD. Declaración de entidades generales internas

Predefinidas en el lenguaje

- <; Se corresponde con el signo menor que, <.
- >; Hace referencia al signo mayor que, >.
- "; Son las comillas rectas dobles, ".
- '; Es el apóstrofe o comilla simple, '.
- &; Es el et o ampersand, &.

Definidas propias

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <!DOCTYPE libros [
3      <!ELEMENT libros (libro)+>
4      <!ELEMENT libro (#PCDATA)>
5
6      <!ENTITY autor "Cervantes">
7      <!ENTITY editorial "Alfaguara">
8  ]>
9
10 <libros>
11   <libro>Don Quijote de la Mancha fue escrito por &autor;</libro>
12   <libro>SIDI fue escrito por Arturo Pérez-Reverte y publicado por &editorial;</libro>
13   <libro>Tiempos recios fue escrito por Mario Vargas Llosa y publicado por &editorial;</libro>
14 </libros>
```

Se hace referencia a la entidad usando &nombre_entidad

Silvia Thomas. Definición entidades internas ([CC0](#))

Al abrir el
fichero en un
navegador, se
vería:

```
<libros>
  <libro>Don Quijote de la Mancha fue escrito por Cervantes</libro>
  <libro>
    SIDI fue escrito por Arturo Pérez-Reverte y publicado por Alfaguara
  </libro>
  <libro>
    Tiempos recios fue escrito por Mario Vargas Llosa y publicado por Alfaguara
  </libro>
</libros>
```

EJERCICIO 2: DTD. Declaración de entidades generales externas

- La declaración de una entidad externa **cuando sólo va a ser utilizada por una única aplicación** es:

```
<!ENTITY nombre_entidad SYSTEM "http://localhost/docxml/fichero_entidad.xml">
```

- La declaración de una entidad externa **cuando va a ser utilizada por varias aplicaciones** es:

```
<!ENTITY nombre_entidad PUBLIC "identificador público formal" "camino hasta la DTD (uri)">
```

EJERCICIO 2: DTD. Declaración de entidades de parámetros

Permiten dar nombres a partes de un DTD y hacer referencia a ellas a lo largo del mismo. Son especialmente útiles cuando varios elementos del DTD comparten listas de atributos o especificaciones de contenidos. Se denotan por % entidad;

Internas

```
<!ENTITY % direccion "calle, numero?, ciudad, cp"> y se puede usar como sigue:  
<!ELEMENT almacen (%direccion;, web)>  
<!ELEMENT oficina (%direccion;, movil)>  
<!ELEMENT central (%direccion;, telefono)>  
<!ELEMENT tienda (%direccion;, fax)>
```

Externas

Permite incluir en un DTD elementos externos, lo que se aplica en dividir la definición DTD en varios documentos.

```
<!ENTITY %persona SYSTEM "persona.dtd">
```

EJERCICIO 2: DTD. Declaración de **elemento terminal**

La declaración de elementos en una DTD (Definición de Tipo de Documento) en XML se realiza utilizando la sintaxis **<!ELEMENT nombre_del_elemento tipo_de_contenido >**. Aquí **tipo_de_contenido** especifica el contenido permitido en el elemento.

Elemento terminal: Aquellos que no contienen más elementos → **tipo_de_contenido**

- **#PCDATA:** Indica que los datos son analizados en busca de etiquetas, resultando que el elemento no puede contener elementos, es decir solo puede contener datos de tipo carácter exceptuando los siguientes: <, [, &,], >.

```
<!ELEMENT ejemplo (#PCDATA)>
```

XML asociado **correcto:**

```
<ejemplo />
```

```
o
```

```
<ejemplo>Esto es un ejemplo</ejemplo>
```

- **EMPTY:** Indica que el elemento no es contenedor, es vacío, es decir, que no puede tener contenido.

```
<!ELEMENT ejemplo EMPTY>
```

XML asociado **correcto:**

```
<ejemplo></ejemplo> ó <ejemplo />
```

- **ANY:** Permite que el contenido del elemento sea cualquier cosa (texto y otros elementos). **No es recomendable usar este tipo de elemento.**

EJERCICIO 2: DTD. Declaración de elemento NO terminal

Estos son los elementos que están formados por otros elementos. Para definirlos utilizamos referencias a los grupos que los componen tal y como muestra el ejemplo: `<!ELEMENT A (B, C)>`. En este caso se ha definido un elemento *A* que está formado por un elemento *B* seguido de un elemento *C*.

Operadores: Además, también hay operadores que nos permiten definir la cardinalidad de un elemento:

- **Operador opción, ?**: Indica que el elemento no es obligatorio.

```
<!ELEMENT telefono (trabajo?, casa )>
```

En el elemento telefono el subelemento trabajo es opcional

- **Operador uno-o-más, +**: Define un componente presente al menos una vez.

```
<!ELEMENT provincia (nombre, (cp, ciudad)+ )>
```

El elemento provincia, está formado por el nombre y otro grupo, que puede aparecer una o varias veces

- **Operador cero-o-mas, ***: Define un componente presente cero, una o varias veces.

```
<!ELEMENT provincia (nombre, (cp, ciudad)* )>
```

El elemento provinicoa, está formado por el nombre y otro grupo, que puede no aparecer o aparecer varias veces

- **Operador de elección, |**: Cuando se utiliza sustituyendo las comas en la declaración de grupos indica que para formar el documento XML hay que elegir entre los elementos separados por este operador.

```
<!ELEMENT provincia (nombre, (cp | ciudad) )>
```

El elemento provincia, está formado por el nombre y el CP o por el nombre y la ciudad

EJERCICIO 2: DTD. Declaración de listas

Mediante las listas vamos a declarar los atributos asociados a un elemento:

```
<! ATTLIST nombre_elemento_del_atributo nombre_atributo tipo_del_atributo "Valor_por_defecto">
```

Si un elemento tiene más de un atributo:

```
<! ATTLIST nombre_elemento_del_atributo1 nombre_atributo1 tipo_del_atributo1 "Valor_por_defecto"
                                     nombre_atributo2 tipo_del_atributo2 "Valor_por_defecto"
.....>
```

Tipos de atributos:

- **Enumeración:** El atributo solo puede tomar uno de los valores determinados dentro de un paréntesis y separados por el operador |.

```
<!ATTLIST fecha dia_semana (lunes|martes|miércoles|jueves|viernes|sábado|domingo) #REQUIRED>
```

- **CDATA:** Se utiliza cuando el atributo es una cadena de texto.

➡ DTD:

```
<!ATTLIST ejemplo color CDATA #REQUIRED>
```

➡ XML asociado:

```
<ejemplo color="verde" />
```

- **ID:** Permite declarar que el valor del atributo (no el nombre) debe ser único y no se puede repetir en otros elementos o atributos.

➡ DTD:

```
<!ATTLIST libro codigo ID #REQUIRED>
```

➡ XML asociado:

```
<libro codigo="Q1">El Quijote</libro>
```

- **IDREF:** Permite hacer referencias a identificadores. En este caso, el valor del atributo ha de corresponder con el de un identificador de un elemento existente en el documento.
- **NMTOKEN:** Permite determinar que el valor de un atributo ha de ser una sola palabra compuesta por los caracteres permitidos por XML, es decir letras, números y los caracteres “:”, “_”, “-” o “.”.

EJERCICIO 2: DTD. Declaración de listas

Tipos de atributos → Como declarar si un atributo es obligatorio o no

- **#IMPLIED**, determina que el atributo sobre el que se aplica es opcional.

• XML asociado válido:

```
<ejemplo color="verde" />
<ejemplo color="" />
<ejemplo/>
```

• DTD:

```
<!ATTLIST ejemplo color CDATA #IMPLIED>
```

- **#REQUIRED**, determina que el atributo tiene carácter obligatorio.

• XML asociado válido:

```
<ejemplo color="" />
<ejemplo color="verde" />
```

• XML asociado no válido:

```
<ejemplo />
```

• DTD:

```
<!ATTLIST ejemplo color CDATA #REQUIRED>
```

- **#FIXED**, permite definir un valor fijo para un atributo, independientemente de que ese atributo se defina explícitamente en una instancia del elemento en el documento XML.

• XML asociado NO válido:

```
<ejemplo color="rojo" />
```

• XML asociado válido:

```
<ejemplo color="verde" />
```

• DTD:

```
<!ATTLIST ejemplo color CDATA #FIXED "verde">
```

- **Literal**, (valor por defecto) asigna por defecto a un atributo el valor dado por una cadena entre comillas, pero puede tomar otro valor

• DTD:

```
<!ATTLIST ejemplo colo (rojo|verde|amarillo) "verde">
```

• XML asociado:

```
<ejemplo color="verde" />
```


EJERCICIO 2: DTD. EJEMPLO RESUELTO

La empresa **Reggio** tiene establecimientos por toda Italia, pero su sede central está en Cesena, al igual que el almacén donde se distribuye a todos los demás establecimientos. Esta empresa distribuye alpiste para pájaros, así como otros artículos ornitológicos.

Cada **establecimiento** tiene una **tienda**, así como un **almacén**, que pueden o no estar en la misma ubicación. Cuando se hace un pedido a la fábrica por parte de los establecimientos, éstas reciben los artículos en su almacén, y la documentación (albarán y pago) se remite a la tienda.

Cada **pedido** debe tener los datos siguientes:

- **Datos del establecimiento** que realiza el pedido (Nombre, dirección para envío y dirección almacén (si es la misma, sólo aparecerá una).
- **Código de pedido** (Cadena de caracteres formada por: Código establecimiento (1 letra y 2 números), número pedido (4 números), un guión y Año (dos números), por ejemplo: E180021-17
- **Nombre del empleado** que realiza el pedido.
- **Fecha** de pedido.
- **Tipo de envío** (cuyos valores son: Normal o Urgente)

Respecto a los **artículos del pedido**, se guardarán los siguientes datos:

- **Código** del artículo (formado por tres letras y 3 números, por ejemplo: ZZZ134.
- Número de **unidades**.
- **Precio por unidad**.
- **Observaciones**.

Herramienta para editar y validar documentos en el lenguaje XML → [XML Copy Editor](#) (Gratuito)

EJERCICIO 2: DTD. EJEMPLO RESUELTO

```
<!--Declaración de elemento no terminal pedido formado por los subelementos (establecimiento, empleado,..., articulos -->
<!ELEMENT pedido (establecimiento,empleado,fecha_pedido,articulos)>
  <!ATTLIST pedido cod_pedido CDATA #REQUIRED> Declaramos una lista en el elemento pedido, llamado, cod_pedido de
  <!ATTLIST pedido tipo (normal|urgente) #REQUIRED> tipo cadena obligatorio
  <!ELEMENT establecimiento (nomb_estab,direccion_envio,direccion_almacen?)>
    <!ELEMENT nomb_estab (#PCDATA)>
    <!ELEMENT direccion_envio (via,numero,localidad,provincia,cp)>
    <!ELEMENT direccion_almacen (via,numero,localidad,provincia,cp)>
      <!ELEMENT via (#PCDATA)> Declaración del elemento terminal via que solo puede tener datos de tipo carácter
      <!ELEMENT numero (#PCDATA)>
      <!ELEMENT localidad (#PCDATA)>
      <!ELEMENT provincia (#PCDATA)>
      <!ELEMENT cp (#PCDATA)>
    <!ELEMENT empleado (#PCDATA)>
    <!ELEMENT fecha_pedido (#PCDATA)> Elemento no terminal articulos que está formado por artículo que
    <!ELEMENT articulos (articulo+)> puede aparecer más de una vez
    <!ELEMENT articulo (unidades,precio,observaciones?)>
      <!ATTLIST articulo cod_articulo CDATA #REQUIRED>
      <!ELEMENT unidades (#PCDATA)>
      <!ELEMENT precio (#PCDATA)>
      <!ATTLIST precio moneda CDATA #REQUIRED>
      <!ELEMENT observaciones (#PCDATA)>
```

tarea04_01.dtd

Vamos a crear el XML y enlazamos este DTD

EJERCICIO 2: DTD. EJEMPLO RESUELTO

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE pedido SYSTEM "tarea04_01.dtd">
<pedido cod_pedido="E180021-17" tipo="urgente">
  <establecimiento>
    <nomb_estab>Manitoba</nomb_estab>
    <direccion_envio>
      <via>Vittorio</via>
      <numero>1</numero>
      <localidad>Cesena</localidad>
      <provincia>Cesena</provincia>
      <cp>00400</cp>
    </direccion_envio>
    <direccion_almacen>
      <via>Vittorio</via>
      <numero>5</numero>
      <localidad>Cesena</localidad>
      <provincia>Cesena</provincia>
      <cp>00400</cp>
    </direccion_almacen>
  </establecimiento>
  <empleado>Roberto Rossini</empleado>
  <fecha_pedido>2017-05-01</fecha_pedido>
  <articulos>
    <articulo cod_articulo="ZZZ134">
      .....
    </articulo>
  </articulos>
</pedido>
```

La solución completa
está disponible en la
plataforma Moodle:

[RETROALIMENTACIÓN COMPLETA](#)

EJERCICIO 3

Generar un XML Schema (Extensión del archivo .xsd)

- ✓ 1º Crearemos el documento XSD: **ejercicio3.xsd**
- ✓ 2º Copiamos 1jercicio1.xml y lo renombraremos el archivo XML creado en el ejercicio 1 como: **ejercicio3_xsd.xml** y lo modificaremos para enlazarlo con **ejercicio3.xsd**
- ✓ **Nota inicial:** Los elementos y atributos XML que se utilizan para generar esquemas pertenecen al espacio de nombres XML Schema, con el URI: <http://www.w3.org/2001/XMLSchema>.
- ✓ Un documento XML Schema tiene como **elemento raíz**, ejemplar, un elemento **<xs:schema>**. Este elemento contendrá las declaraciones de todos los elementos y atributos que puedan aparecer en un documento XML asociado. Este elemento tendrá un elemento hijo que será el elemento raíz del documento XML, que declararemos con **<xs:element>**.
- ✓ Por tanto, nuestro documento .xsd deberá comenzar:

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