DTD

LENGUAJES DE MARCAS Y SISTEMAS DE GESTIÓN DE INFORMACIÓN

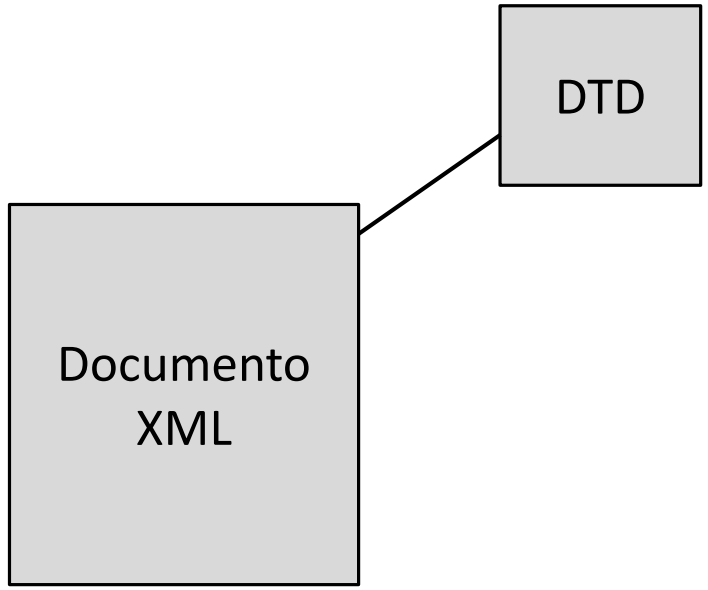
Introducción / Recordatorio

- Cuando un documento XML no tiene DTD, cualquier etiqueta que contenga se considerará válida. Sólo se puede comprobar que el documento esté bien formado.
- Sin embargo, es muy recomendable utilizar un **DTD** que defina formalmente el lenguaje de etiquetado (qué elementos/etiquetas hay, qué puede contener cada uno…). El DTD garantiza que los documentos sigan una serie de reglas. Será entonces cuando, además de bien formado, las etiquetas tendrán que ser válidas (basándonos en el DTD).
- De hecho, crear el DTD es el primer paso antes de escribir el documento XML

  (al igual que en una BD, por ejemplo, 1º definimos el esquema con el que luego se trabaja)

Introducción / Recordatorio

Todas las etiquetas son válidas → Documento válido
¿Documento bien formado?



¿Etiquetas válidas según el DTD? → ¿Documento válido?
¿Documento bien formado?

Declaración del DTD

- El DTD se declara debajo de la declaración XML (o sea, en la 2ª línea del XML).
- Varias opciones
  - Incluir en el mismo XML el contenido del DTD

```
<!DOCTYPE elemento_raiz [
  Aquí todo el contenido del DTD
]>
```

  - Referencia a un archivo externo (lo normal, ya que un DTD se suele usar para varios XML)
    - DTD no publicado

```
<!DOCTYPE elemento_raiz SYSTEM "archivo.dtd">
```

    - DTD publicado

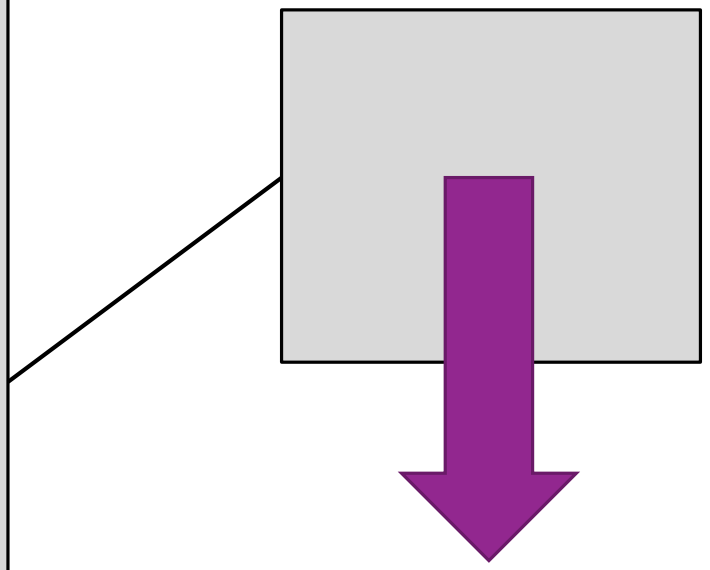
```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML
4.01//EN" "http://www.w3.org/TR/html4/strict.dtd">
```

Declaración del DTD

biblioteca.xml

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE catalogo SYSTEM "midtd.dtd">
<catalogo>
        <libro>
                <titulo>Don Quijote</titulo>
                <autor nacionalidad="español">Cervantes</autor>
                <genero>Aventuras</genero>
        </libro>
        <libro>
                <titulo>Los pilares de la tierra</titulo>
                <autor nacionalidad="británico">Ken Follet</autor>
                <genero>Histórica</genero>
        </libro>
</catalogo>
```

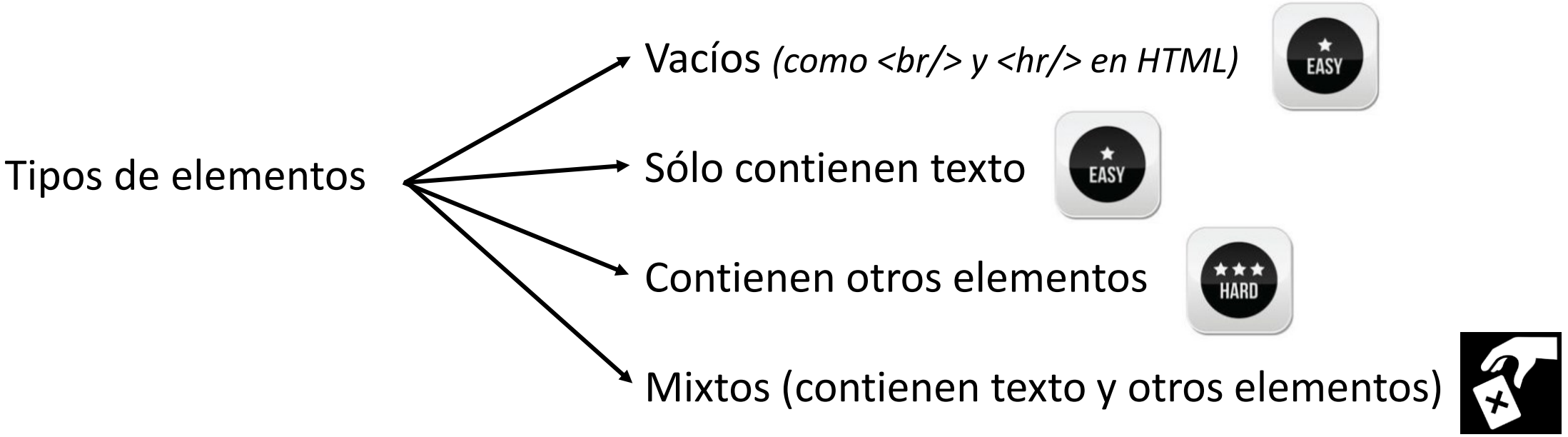
midtd.dtd



Ahora veremos todo lo que hay que poner en el DTD

DTD → Declarar elementos

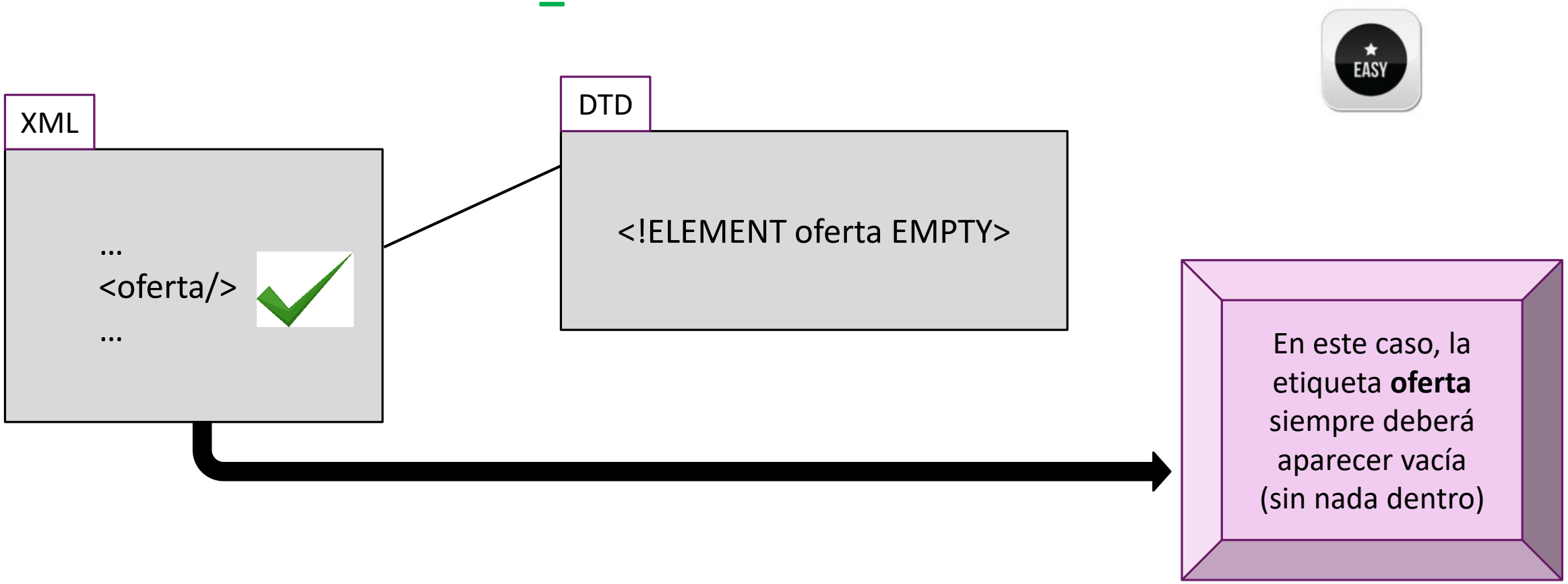
- Si recordamos, hay 4 tipos de elementos:



- Ahora veremos cómo se declara en un DTD cada uno de ellos…

DTD → Declarar elementos → Vacíos

<!ELEMENT nombre_elemento EMPTY>



DTD → Declarar elementos

<!ELEMENT nombre_elemento (#PCDATA)>

XML

```
...
<apellidos>Navarro Ruiz</apellidos>
...
```

DTD

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

En este caso, la etiqueta **apellidos** siempre deberá aparecer conteniendo un texto

DTD → Declarar elementos

- **SECUENCIALES:** **<!ELEMENT nombre_elemento (uno,dos,tres)>**

  El elemento **nombre_elemento** contiene los elementos **uno**, **dos** y **tres** en ese orden.

  Por ejemplo → **<!ELEMENT mensaje (destinatario, asunto, cuerpo)>**

XML

```
…
<mensaje>
  <destinatario>…</destinatario>
  <asunto>…</asunto>
  <cuerpo>…</cuerpo>
</mensaje>
…
```

- **ALTERNATIVOS:** **<!ELEMENT nombre_elemento (uno|dos|tres)>**

  El elemento **nombre_elemento** contiene sólo uno de los elementos hijos.

  Por ejemplo → **<!ELEMENT trabajador (activo | parado | baja)>**

XML

```
…
<trabajador>
  <activo/>
</trabajador>
…
```

XML

```
…
<trabajador>
  <parado/>
</trabajador>
…
```

XML

```
…
<trabajador>
  <baja/>
</trabajador>
…
```

También se podrían combinar ambas cosas:

separación con **,** *y con* **|**

DTD → Declarar elementos

…y, además, se puede añadir un carácter a la derecha de cada hijo para saber cuántas veces puede aparecer:

- Si no se añade nada → debe aparecer **una vez**
- Si se añade * → puede repetirse **0 o más veces**
- Si se añade ? → puede aparecer **0 o 1 vez**
- Si se añade + → puede repetirse **1 o más veces**

DTD

```
<!ELEMENT club
(nombre,titulo*,mascota?,color+)>
```

XML

```
<club>
  <nombre>Madrid</nombre>
  <titulo>Liga</titulo>
  <titulo>Champions</titulo>
  <titulo>Copa</titulo>
  <color>Blanco</color>
</club>
```

XML

```
<club>
  <nombre>Villabotijo</nombre>
  <mascota>Paquetonic</mascota>
  <color>Negro</color>
  <color>Amarillo</color>
</club>
```

XML

```
<club>
  <nombre>Cifuentes FC</nombre>
  <mascota>Master</mascota>
  <mascota>Botes de crema</mascota>
</club>
```

DTD → Declarar elementos → Mixtos

<!ELEMENT nombre_elemento (#PCDATA|uno|dos|tres)*>

Este caso **no suele utilizarse en XML**, por lo que **lo descartaremos**.

Dicha declaración tendría, además, limitaciones. No hay ningún control sobre la frecuencia de aparición de los hijos. El formato sería algo rígido.

Por ello, nos quedaremos con los 3 casos vistos antes

- Elementos vacíos
- Elementos que contienen texto
- Elementos que contienen otros elementos

DTD → Ejemplo

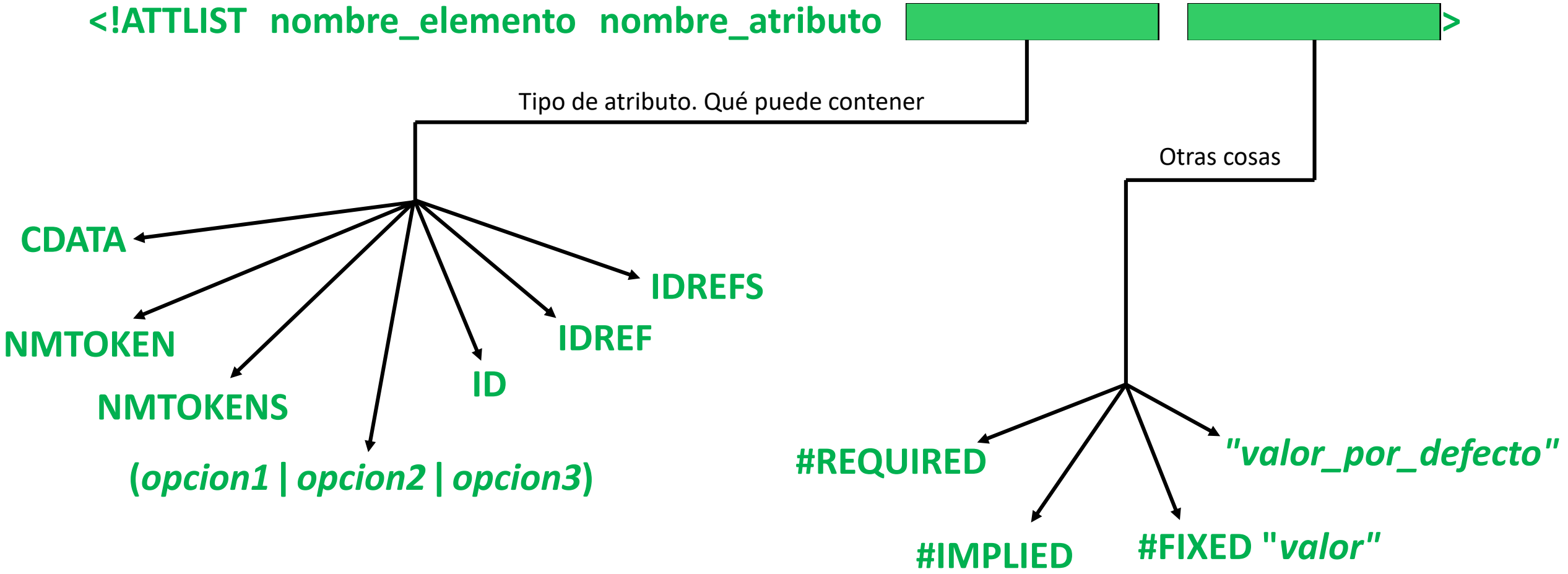
midtd.dtd

```
<!ELEMENT agenda (contacto*)>
<!ELEMENT contacto (nombre,apellido+,telefono,email?,direccion)>

<!ELEMENT nombre (#PCDATA)>
<!ELEMENT apellido (#PCDATA)>
<!ELEMENT telefono (#PCDATA)>
<!ELEMENT email (#PCDATA)>

<!ELEMENT direccion (calle,numero,cp)>
<!ELEMENT calle (#PCDATA)>
<!ELEMENT numero (#PCDATA)>
<!ELEMENT cp (#PCDATA)>
```

Así podría quedar un archivo DTD. En este caso:

- El elemento que engloba a todos es **agenda**
- El elemento agenda puede contener 0 o más elementos de tipo **contacto**
- Dentro de contacto habrá un elemento **nombre**, 1 o más elementos **apellido**, un **telefono**, 0 o 1 elementos **email** y un elemento **direccion**. Todos ellos contienen texto en su interior.
- Dentro de direccion habrá un elemento **calle**, un elemento **numero** y un elemento **cp**. Todos ellos contienen texto en su interior.

DTD → Declarar atributos



DTD → Declarar atributos → Tipos

Después de **<!ATTLIST nombre_elemento nombre_atributo** hay que indicar el tipo de atributo (qué puede contener dicho atributo). Hay 7 opciones:

- **CDATA**
- **NMTOKEN**
- **NMTOKENS**
- Atributos enumerados → (***opcion1*** | ***opcion2*** | ***opcion3***)
- **ID**
- **IDREF**
- **IDREFS**

A continuación veremos qué significa cada uno de ellas…

DTD → Declarar atributos → Tipos

CDATA → cadena de caracteres, incluidos los espacios en blanco

XML

```
...
<autor pais="Estados Unidos">Henry James</autor>
...
```

DTD

```
<!ELEMENT autor (#PCDATA)>
<!ATTLIST autor pais CDATA>
```

DTD → Declarar atributos → Tipos

NMTOKEN → texto, pero permitiendo sólo los caracteres con los que se pueden definir atributos y elementos

El nombre de un elemento, para ser válido, debe empezar por letra o _ y luego puede tener letras, dígitos, ., _ o -. No puede empezar por XML o xml.

XML

```
...
<autor pais="Estados Unidos">Henry James</autor>
...
```

XML

```
...
<autor pais="EEUU">Henry James</autor>
...
```

DTD

```
<!ELEMENT autor (#PCDATA)>
<!ATTLIST autor pais NMTOKEN>
```

[Lo simplificaremos diciendo que NMTOKEN permite una cadena de caracteres sin espacios]

DTD → Declarar atributos → Tipos

NMTOKENS → lista de varios NMTOKEN, separados por espacios

XML

```
...
<coche color="azul rojo amarillo">
        <marca>Kia</marca>
        <modelo>Stonic</modelo>
</coche>
...
```

DTD

```
<!ELEMENT coche (marca,modelo)>
<!ELEMENT marca (#PCDATA)>
<!ELEMENT modelo (#PCDATA)>
<!ATTLIST coche color NMTOKENS>
```

DTD → Declarar atributos → Tipos

(opcion1 | opcion2 | …) → **El contenido del atributo queda restringido a una serie de opciones. Otro valor no sería válido. Se llaman atributos enumerados.**

XML

```
…
<persona sexo="varón">Guillermo</persona>
…
```

XML

```
…
<persona sexo="poco">Bartolo</persona>
…
```

DTD

```
<!ELEMENT persona (#PCDATA)>
<!ATTLIST persona sexo (varón | mujer)>
```

[Pueden ponerse más de 2 opciones]

DTD → Declarar atributos → Tipos

ID → Como un NMTOKEN (teniendo que empezar por letra), con el añadido de que no podrá haber dos elementos con el mismo contenido en ese atributo

XML

```
…
<producto codigo="P123456789"/>
<producto codigo="P555555555"/>
<producto codigo="P000000000"/>
…
```

XML

```
…
<producto codigo="P123456789"/>
<producto codigo="P555555555"/>
<producto codigo="P123456789"/>
…
```

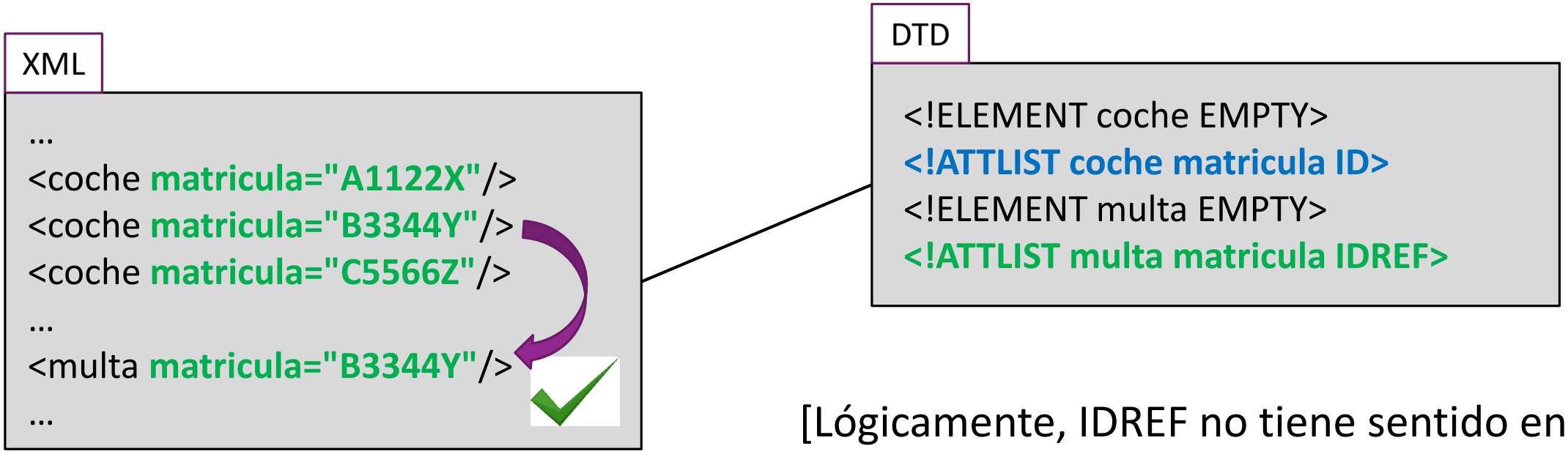
DTD

```
<!ELEMENT producto EMPTY>
<!ATTLIST producto codigo ID>
```

[Ojo al inicio por letra, si se usa un atributo de tipo ID para DNI, matrícula de coche…]

DTD → Declarar atributos → Tipos

IDREF → El contenido debe ser el mismo que el de otro atributo ID incluído en el documento (o sea, IDREF es una referencia a un atributo ID)



[Lógicamente, IDREF no tiene sentido en un DTD si no hay algún atributo ID]

DTD → Declarar atributos → Tipos

IDREFS → Como IDREF, pero se pueden poner varios, separados por espacios

XML

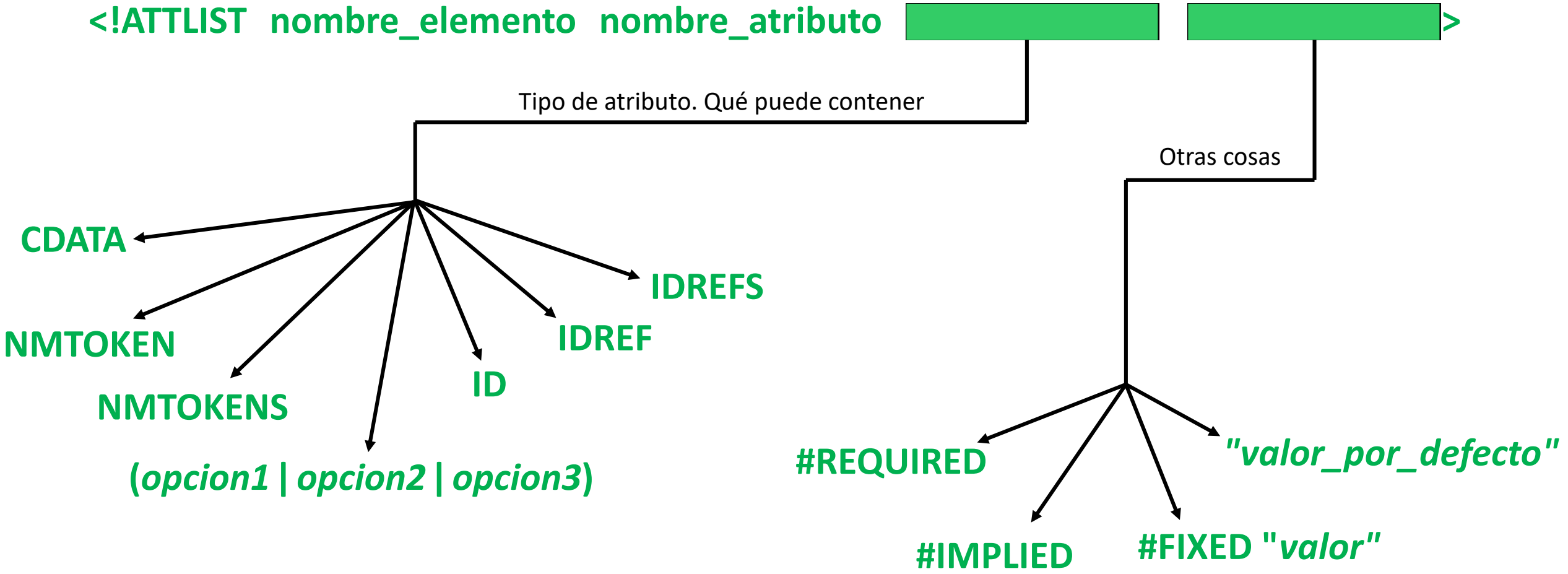
```
...
<coche matricula="A1122X"/>
<coche matricula="B3344Y"/>
<coche matricula="C5566Z"/>
...
<persona matricula="A1122X  B3344Y"/>
...
```

DTD

```
<!ELEMENT coche EMPTY>
<!ATTLIST coche matricula ID>
<!ELEMENT persona EMPTY>
<!ATTLIST persona matricula IDREFS>
```

[IDREFS tampoco tiene sentido
si no hay algún atributo ID]

DTD → Declarar atributos



DTD → Declarar atributos → 2º parámetro

Después de **<!ATTLIST nombre_elemento nombre_atributo** y el **tipo**, hay un segundo parámetro que indica otras cosas del atributo. Hay 4 opciones:

- **#REQUIRED**
- **#IMPLIED**
- **#FIXED *"valor"***
- ***"valor_por_defecto"***

A continuación veremos qué significa cada uno de ellas…

DTD → Declarar atributos → 2º parámetro

#REQUIRED → El atributo es <u>obligatorio</u>

XML

```
...
<autor pais="Estados Unidos">Henry James</autor>
...
```

XML

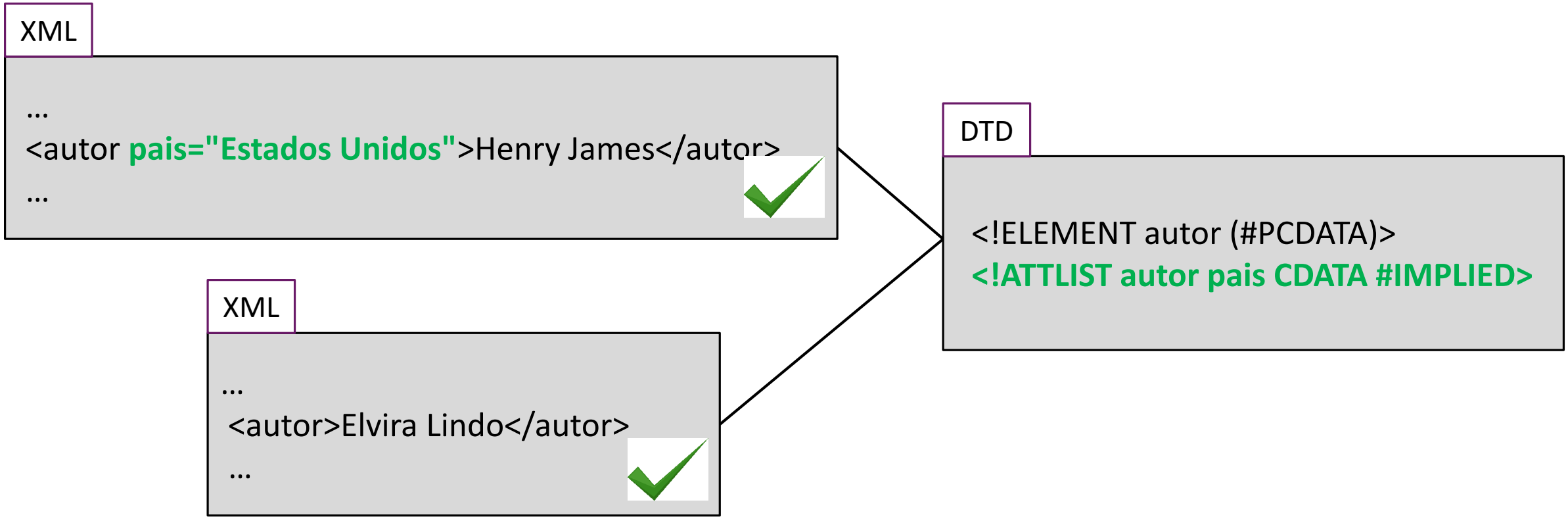
```
...
 <autor>Elvira Lindo</autor>
 ...
```

DTD

```
<!ELEMENT autor (#PCDATA)>
<!ATTLIST autor pais CDATA #REQUIRED>
```

DTD → Declarar atributos → 2º parámetro

#IMPLIED → El atributo es <u>opcional</u>



DTD → Declarar atributos → 2º parámetro

#FIXED *"valor"* → El atributo es obligatorio y tiene un valor fijo que indicamos después

XML

```
…
<coche oferta="sí">
        <marca>Kia</marca>
        <modelo>Stonic</modelo>
</coche>
…
```

XML

```
…
<coche oferta="10%">
        <marca>Kia</marca>
        <modelo>Stonic</modelo>
</coche>
…
```

DTD

```
<!ELEMENT coche (marca,modelo)>
<!ELEMENT marca (#PCDATA)>
<!ELEMENT modelo (#PCDATA)>
<!ATTLIST coche oferta CDATA #FIXED="sí">
```

DTD → Declarar atributos → 2º parámetro

"valor_por_defecto" **→ Si no se declara, se entenderá que el atributo vale lo que digamos**

XML

```
...
<politico tipo="honrado">
        <nombre>Pepe</nombre>
</politico>
...
```

XML

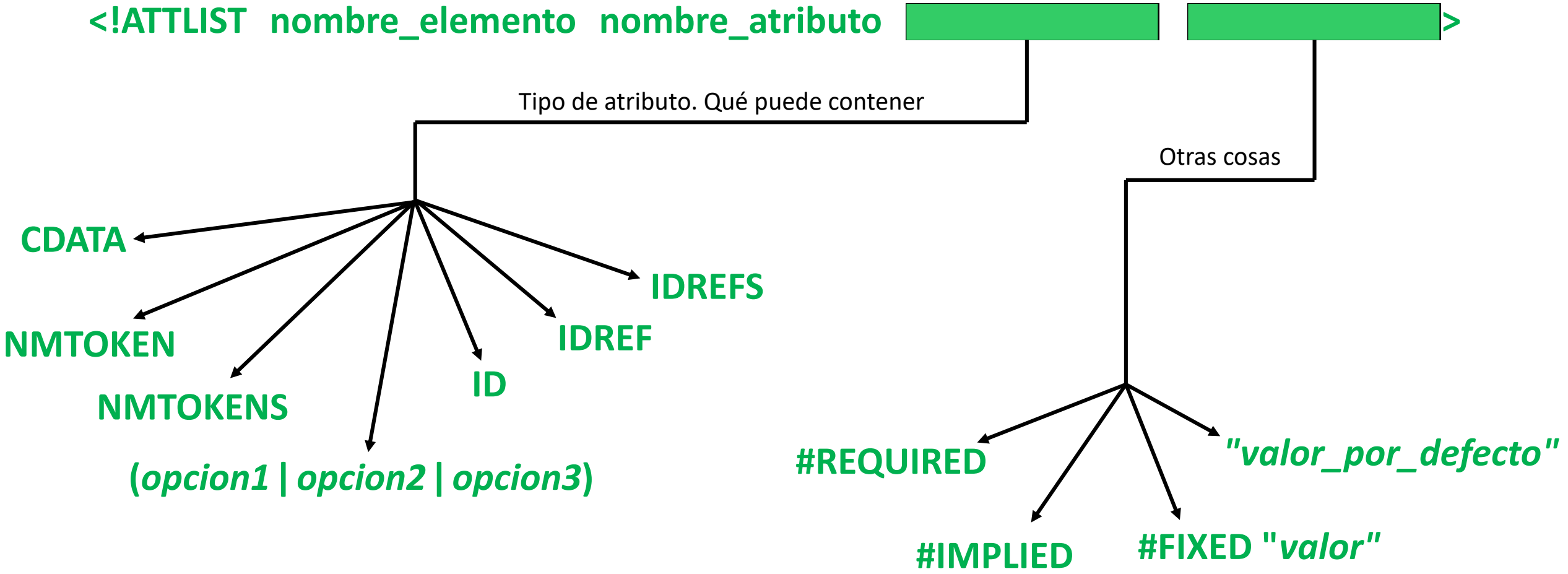
```
...
<politico>
        <nombre>Juan</nombre>
</politico>
...
```

DTD

```
<!ELEMENT politico (nombre)>
<!ELEMENT nombre (#PCDATA)>
<!ATTLIST politico tipo CDATA "corrupto">
```

[Aunque no lo especifiquemos,
Juan es de tipo corrupto]

DTD → Declarar atributos



DTD → Declarar atributos

DTD

```
<!ELEMENT producto (#PCDATA)>
<!ATTLIST producto codigo ID #REQUIRED>
<!ATTLIST producto pais NMTOKEN "España">
<!ATTLIST producto material (plástico|madera) #IMPLIED>
```

=

DTD

```
<!ELEMENT producto (#PCDATA)>
<!ATTLIST producto
   codigo ID #REQUIRED
   pais NMTOKEN "España"
   material (plástico|madera) #IMPLIED>
```

[Sólo cuando sean varios atributos para el **mismo elemento**]

XML

```
…
<producto codigo="P01" pais="EEUU">Televisión americana</producto>
<producto codigo="P02" material="madera">Mueble de IKEA</producto>
…
```

DTD → Declarar entidades

- Una **entidad** es un objeto utilizado para guardar información. Permite guardar contenido que puede ser utilizado muchas veces y poder descomponer un documento grande en subconjuntos más manejables.

- Dos tipos
  - Entidad interna
    - Abreviatura definida en el DTD.
    - Ejemplo: **<!ENTITY derechos "Copyleft 2018">**
    - Luego en el XML bastaría con poner **&derechos;**
  - Entidad externa
    - Se hace referencia a un contenido que está fuera del DTD (internet o local). Esto permite descomponer archivos en unidades más pequeñas.
    - Ejemplo: **<!ENTITY intro SYSTEM "http://ww.servidor.com/intro.xml">**