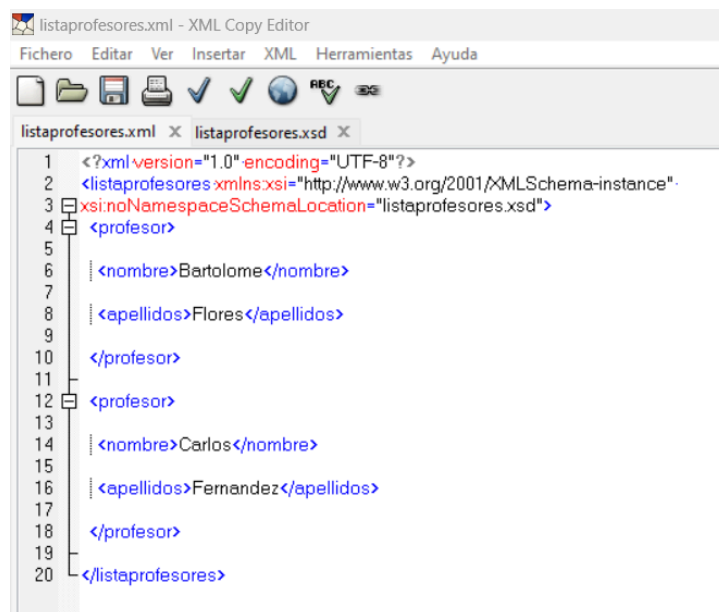


## Lenguajes de marcas

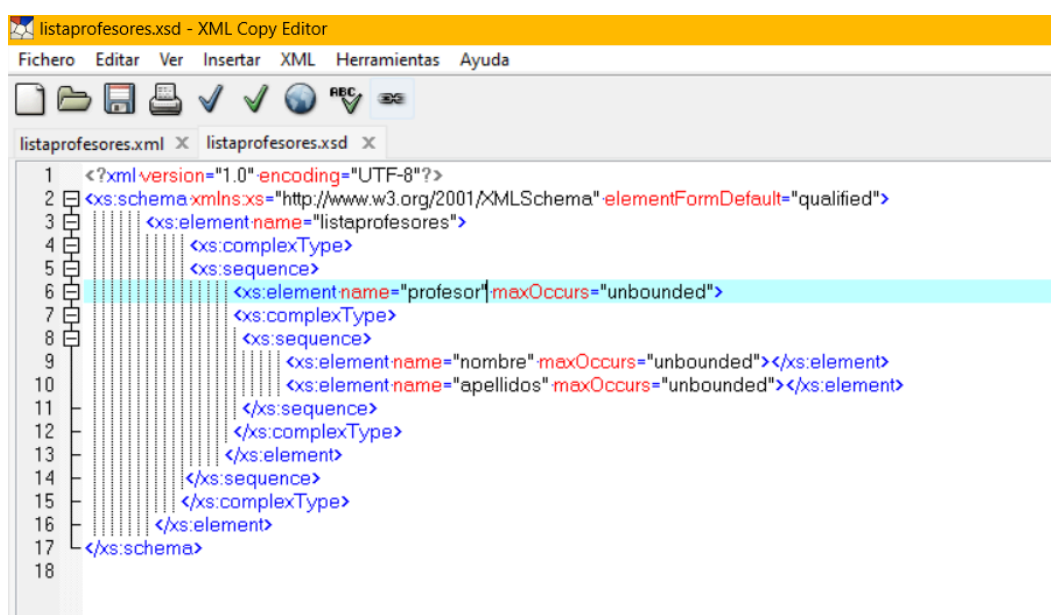
Tarea Tema 11: XML y  
validación XSD

Ejercicio 1: a partir del siguiente xml, se ha creado el xsd.



```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <listaprofesores xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3   xsi:noNamespaceSchemaLocation="listaprofesores.xsd">
4   <profesor>
5     <nombre>Bartolome</nombre>
6     <apellidos>Flores</apellidos>
7   </profesor>
8   <profesor>
9     <nombre>Carlos</nombre>
10    <apellidos>Fernandez</apellidos>
11  </profesor>
12 </listaprofesores>
```

Se ha comentado en el .xsd lo usado y porqué. Como resumen, al tener un elemento raíz llamado listaprofesores, se ha especificado primero este elemento en el xsd, seguido contiene más elementos que serán profesor, por lo que ponemos que es de tipo complejo, al igual que después especificamos lo mismo para profesor que va a contener nombre y apellidos. Todos los elementos se podrán crear de forma ilimitada (unbounded). Además con sequence detallamos que tienen que seguir un orden específico.



```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified">
3   <xs:element name="listaprofesores">
4     <xs:complexType>
5       <xs:sequence>
6         <xs:element name="profesor" maxOccurs="unbounded">
7           <xs:complexType>
8             <xs:sequence>
9               <xs:element name="nombre" maxOccurs="unbounded"></xs:element>
10              <xs:element name="apellidos" maxOccurs="unbounded"></xs:element>
11            </xs:sequence>
12          </xs:complexType>
13        </xs:element>
14      </xs:sequence>
15    </xs:complexType>
16  </xs:element>
17 </xs:schema>
```



Ejercicio 2: a partir del .xsd, creamos 5 alumnos.

En el .xsd encontramos primero un elemento raíz que es practicas, de tipo complejo y siguiendo un orden, por lo que en el xml lo primero que crearemos será prácticas. Después vemos otro elemento que es alumno, también complejo y que va a contener los elementos nif y resultado. Por lo que en el xml, ya tendríamos creado prácticas, con alumno y dentro de alumno, nif y resultado. Finalmente, vemos que cada uno de estos últimos elementos contienen un tipo personalizado, donde vamos a ver las restricciones que tienen para poder rellenar esos campos. En este caso en NIF escribir un patrón de 8 dígitos y una letra de la A a la Z, tanto mayúscula como minúscula y por último en Resultado, solo podemos poner los valores Apto o No Apto, si escribimos algo diferente, no podrá validar correctamente el xml

```
listaprofesores.xml x listaprofesores.xsd x practicas.xml x practicas.xsd x discsduros.xml x discsduros.xsd x
1 <?xml version="1.0" encoding="UTF-8"?>
2 <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
3
4   <!-- se especifica el elemento raíz que es practicas -->
5   <xs:element name="practicas">
6     <!-- va a ser de tipo complejo porque contiene más elementos dentro de otros -->
7     <xs:complexType>
8       <!-- los elementos dentro de practicas deben aparecer en un orden específico. -->
9       <xs:sequence>
10        <!-- después tenemos el elemento alumno, que se va a poder repetir un número ilimitado de veces (unbounded) -->
11        <xs:element name="alumno" maxOccurs="unbounded">
12          <!-- como vamos a tener otros elementos dentro de alumno, ponemos de que tipo serán, en este caso, complejo -->
13          <xs:complexType>
14            <xs:sequence>
15              <!-- elemento nif que es de tipo personalizado denominado tipoNIF -->
16              <xs:element name="nif" type="tipoNIF"/>
17              <!-- elemento resultado, que es de tipo personalizado llamado tipoResultado -->
18              <xs:element name="resultado" type="tipoResultado"/>
19            </xs:sequence>
20          </xs:complexType>
21        </xs:element>
22      </xs:sequence>
23    </xs:complexType>
24  </xs:element>
25
26  <!-- TIPOS PERSONALIZADOS -->
27
28  <!-- primero encontramos un tipo "tipoNIF" de tipo simple -->
29  <xs:simpleType name="tipoNIF">
30    <!-- la restricción es que tiene que ser string, es decir, una cadena de caracteres -->
31    <xs:restriction base="xs:string">
32      <!-- especifica que el valor tiene que seguir un patrón de 8 dígitos y letras en mayúscula o minúscula de la A a la Z -->
33      <xs:pattern value="\d{8}[a-zA-Z]"/>
34    </xs:restriction>
35  </xs:simpleType>
36
37  <!-- el segundo tipo personalizado es "tipoResultado" de tipo simple -->
38  <xs:simpleType name="tipoResultado">
39    <!-- la restricción es que tiene que contener un string, una cadena de caracteres -->
40    <xs:restriction base="xs:string">
41      <!-- lo siguiente restringe los valores permitidos y puede ser solo Apto o No Apto lo que contenga ese elemento -->
42      <xs:enumeration value="Apto"/>
43      <xs:enumeration value="No Apto"/>
44    </xs:restriction>
45  </xs:simpleType>
46
47  </xs:schema>
```



```
1  <?xml version="1.0" encoding="UTF-8"?>
2
3  <practicas xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4  xsi:noNamespaceSchemaLocation="practicas.xsd">
5    <alumno>
6      <nif>77777777H</nif>
7      <resultado>Apto</resultado>
8    </alumno>
9
10   <alumno>
11     <nif>88888888W</nif>
12     <resultado>Apto</resultado>
13   </alumno>
14
15   <alumno>
16     <nif>45555555E</nif>
17     <resultado>No Apto</resultado>
18   </alumno>
19
20   <alumno>
21     <nif>36411111F</nif>
22     <resultado>No Apto</resultado>
23   </alumno>
24
25   <alumno>
26     <nif>39176458D</nif>
27     <resultado>Apto</resultado>
28   </alumno>
29 </practicas>
```

### Ejercicio 3: Crear archivo .xml y .xsd a LIBRE ELECCIÓN

Se ha creado un .xml y .xsd basado en un almacen de discos duros, donde guardamos datos de la marca, modelo y especificaciones (capacidad y número de serie)

El .xsd se ha formado de la siguiente manera:

- Primero tenemos un elemento raíz que es almacenDiscos, de tipo complejo y que va a seguir un orden específico
- este elemento contendrá discos, que también será de tipo complejo y seguirá un orden
- dentro de discos, tenemos los elementos: marca, modelo y especificaciones
- en especificaciones añadiremos dos elementos más por lo que, detallamos el tipo que será, que es complejo porque son más de un elemento y al final lo mismo que los anteriores, sigue un orden



- para limitar lo que es el número de serie, creamos un tipo personalizado llamado numSerie, donde limitaremos los valores para que sigan un patron que es comenzar por WD o SG y tener 9 digitos (en este caso, también se podría poner directamente SN porque todos los discos lo contienen pero para simplificar)

Queda el siguiente .xml siguiendo las reglas dichas anteriormente:

```
listaprofesores.xml X listaprofesores.xsd X practicas.xml X practicas.xsd X discosduros
1 <?xml version="1.0" encoding="UTF-8"?>
2 <almacenDiscos xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3   xsi:noNamespaceSchemaLocation="discosduros.xsd">
4
5   <disco>
6     <marca>Western Digital</marca>
7     <modelo>WD-Blue</modelo>
8     <especificaciones>
9       <capacidad>1000</capacidad>
10      <numSerie>WD123456789</numSerie>
11    </especificaciones>
12  </disco>
13
14  <disco>
15    <marca>Seagate</marca>
16    <modelo>Barracuda</modelo>
17    <especificaciones>
18      <capacidad>2000</capacidad>
19      <numSerie>SG987654321</numSerie>
20    </especificaciones>
21  </disco>
22
23 </almacenDiscos>
```