1. Qué es un esquema XSD

Los DTD permiten diseñar un vocabulario para ficheros XML, pero, ¿qué sucede cuando los valores de los elementos y atributos de esos ficheros han de corresponder a datos de un tipo determinado, o cumplir determinadas restricciones que no pueden reflejarse en los DTD? Para ello se definen **XML Schemas**. Un XML Schema, al igual que las DTD, es un mecanismo para comprobar la validez de un documento XML. Se trata de una forma alternativa a las DTD, pero con bastantes *ventajas* sobre estos:

- Es un documento XML, por lo que se puede comprobar si está bien formado.
- Existe una extensa lista de tipos de datos predefinidos para elementos y atributos que pueden ser ampliados o restringidos para crear nuevos tipos.
- Permiten concretar con precisión la cardinalidad de un elemento, es decir, las veces que puede aparecer en un documento XML.
- Permite mezclar distintos vocabularios gracias a los espacios de nombres.

Pero también tienen alguna *desventaja*:

- Son documentos más difíciles de programar e interpretar.

Cuando creamos un Schema creamos un nuevo archivo y lo guardamos con extensión .xsd motivo por el cual también se les denomina documentos XSD.

Ejercicio resuelto

Creación de un esquema correspondiente al siguiente documento XML:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes?>
<!DOCTYPE alumno>
<alumno edad="22">Olga Velarde Cobo</alumno>
```

Ocultar Información

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

<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
   <xs:element name="alumno" type="xs:string"/>
</xs:schema>
```

2. Qué es un esquema XSD

Un **esquema XSD** es un documento XML con la siguiente estructura:

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

- La etiqueta **xml** define la versión de XML utilizada y la codificación de caracteres del documento.
- La etiqueta raíz **xs:schema** contiene declaraciones para todos los elementos y atributos que puedan aparecer en un documento XML. Dentro se define el espacio de nombres o namespace del Schema. En informática un espacio de nombres es un vocabulario controlado donde todos los elementos se identifican de forma única. La declaración de un namespace en XML sigue la siguiente sintaxis **xmlns:xs="URI"** donde URI es la identificación del namepace mediante su dirección URL. En un Schema se pueden definir otros namespaces, ya sea dentro del XSD mediante el atributo targetNamespace dentro del xs:schema o en el propio XML usando el atributo xmlns dentro de la etiqueta raíz y usando en las etiquetas el prefijo que identificada a cada namespace.

XSD

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

XML

```
<paises xmlns:e="http://www.prueba.es/economia"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="paises.xsd">
 <pais>España<capital>Madrid</capital><e:capital>30.370.89$</e:capital>
```

En el XML tenemos 2 etiquetas con el mismo nombre pero se identifican de forma única porque pertenecen a namespaces distintos.

El **documento XML** deberá referenciar el esquema XSD con el que se va a validar utilizando la siguiente cabecera:

```
<?xml version="1.0" encoding="UTF-8"?>
<alumnos xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="alumnos.xsd"></alumnos>
```

- **xmlns:xsi**: para declarar el espacio de nombres del esquema XSD.
- **xsi:noNamespaceSchemaLocation**: para vincular el documento XML con el esquema local XSD.

3. Componentes básicos de un esquema XSD

Los componentes imprescindibles para construir un esquema XSD son los siguientes:

- **xs:element**: permite declarar las etiquetas del documento XML. Los elementos en un Schema pueden ser de 2 tipos:
  - **Elementos simples**: son etiqueta XML que sólo contienen datos. Se identifican en Schema como **<xs:simpleType>**
  - **Elementos compuestos**: son etiqueta XML que incluyen dentro otros elementos y/o atributos. Hay 4 clases de elementos complejos. Se identifican en Schema como **<xs:complexType>**
    - Elementos vacíos.
    - Elementos que contienen otros elementos.
    - Elementos mixtos: otros elementos y datos
    - Cada uno de los elementos anteriores pueden contener atributos
    - Elementos que contienen sólo atributos.

- **xs:attribute**: permite declarar los atributos que incluyen las etiquetas del documento XML

4. xs:element

Los principales atributos que podemos utilizar en la declaración son los siguientes:

- ***name***: Indica el nombre del elemento. Obligatorio si el elemento padre es *<xs:schema>*.
- ***type***: Indica el tipo de dato que almacenará el elemento. No puede aparecer junto con *ref*.
- ***default*:** Es el valor que tomará el elemento al ser procesado si en el documento XML no ha recibido ningún valor.
- ***fixed*:** Indica el único valor que puede contener el elemento en el documento XML.
- ***minOccurs*:** Indica el mínimo número de ocurrencias que deben aparecer de ese elemento en el documento XML. No se puede usar si el elemento padre es *<xs:schema>*. Va desde 0 hasta ilimitado (*unbounded*). Por defecto 1.
- ***maxOccurs*:** Indica el máximo número de ocurrencias que pueden aparecer de ese elemento en el documento XML. No se puede usar si el elemento padre es *<xs:schema>*. Va desde 0 hasta ilimitado (*unbounded*). Por defecto 1.
- ***ref***: Indica que la declaración del elemento se encuentra en otro lugar del esquema. No se puede usar si el elemento padre es *<xs:schema>*. No puede aparecer junto con *name*.

Ejemplo de XML:

```
<nombre>Jana</nombre>
<edad>36</edad>
<fechanacimiento>1970-03-27</fechanacimiento>
```

Ejemplo de XML Schema:

```
<xs:element name="nombre" type="xs:string"/>
<xs:element name="edad" type="xs:integer"/>
<xs:element name="fechanacimiento" type="xs:date"/>
```

EJEMPLOS:

```
<xs:element name="nombre" type="xs:string" default="TuNombre" minOccurs="1" maxOccurs="10" />
<xs:element name="edad"  type="xs:integer" fixed="25" />
<xs:element ref="contacto" minOccurs="1" maxOccurs="unbounded" />
```

5. xs:attribute

Este componente permite declarar los atributos de los elementos del documento XML. Entre otros, los principales atributos que podemos utilizar en la declaración son los siguientes:

- ***name***: Indica el nombre del atributo.
- ***ref***: Indica que la declaración del atributo se encuentra en otro lugar del esquema. No puede aparecer junto con *name*.
- ***type***: Indica el tipo de dato que almacenará el atributo. No puede aparecer junto con *ref*.
- ***use***: Indica si la existencia del atributo es opcional (*optional*), obligatoria (*required*) o prohibida (*prohibited*). Por defecto opcional.
- ***default***: Es el valor que tomará el elemento al ser procesado si en el documento XML no ha recibido ningún valor.
- ***fixed***: Indica el único valor que puede contener el elemento en el documento XML. Sólo se puede usar con tipo de dato textual.

```
<xs:attribute name="moneda" type="xs:string" default="euro" use="required" />
```

6. Elementos simples <xs:simpleType>

a. **Tipos de datos predefinidos:** Son los distintos valores que puede tomar el **atributo type** cuando se declara un elemento. Algunos de los principales valores predefinidos son:
   - **String**: se corresponde con una cadena de caracteres UNICODE.
   - **Boolean**: representa valores lógicos, es decir que solo pueden tomar dos valores, true o false.
   - **Integer**: número entero positivo o negativo.
   - **positiveInteger**: número entero positivo.
   - **negativeInteger**: número entero negativo.
   - **Decimal**: número decimal, por ejemplo, 8,97.
   - **time**, hora en el formato hh:mm:ss.
   - **date**, fecha en formato CCYY-MM-DD.
   - **ID, IDREF, ENTITY, NOTATION, MTOKEN**. Representan lo mismo que en los DTD

b. **Tipos de datos construidos:** son generados por el usuario a partir de un tipo de dato predefinido y aplicándoles restricciones (o también llamadas facetas). Las restricciones puedes usarse **directamente en la definición de un elemento**, en lugar de usar el atributo *type*

```
<xs:element name="edad" minOccurs="1" maxOccurs="1">
 <xs:simpleType>
  <xs:restriction base="xs:integer">
   <xs:minInclusive value="0"/>
   <xs:maxInclusive value="100"/>
  </xs:restriction>
 </xs:simpleType>
</xs:element>
```

6.1. Restricciones (facetas)

Solo pueden aplicarse sobre tipos simples utilizando el elemento <**xs:restriction>**. Se expresan como un elemento dentro de una restricción y se pueden combinar para lograr restringir más el valor del elemento

- **Rango de números**: Se utiliza en los tipos de datos numéricos y de fecha/hora. Define el mínimo y máximo, tanto inclusive como exclusive, de los números permitidos.

```
<xs:element name="edad" minOccurs="1" maxOccurs="1">
 <xs:simpleType>
  <xs:restriction base="xs:integer">
   <xs:minInclusive value="0"/>
   <xs:maxInclusive value="100"/>
  </xs:restriction>
 </xs:simpleType>
</xs:element>
```

- **Dígitos**: Se utiliza en los tipos de datos numéricos. Define el número máximo de dígitos permitidos.

```
<xs:element name="telefono" minOccurs="1" maxOccurs="1">
 <xs:simpleType>
  <xs:restriction base="xs:integer">
   <xs:totalDigits value="9"/>
  </xs:restriction>
 </xs:simpleType>
</xs:element>
```

- **Lista de valores**: Se utiliza en todos los tipos de datos. Define una lista de valores permitidos en el elemento.

```
<xs:element name="coche" minOccurs="1" maxOccurs="1">
 <xs:simpleType>
  <xs:restriction base="xs:string">
   <xs:enumeration value="Audi"/>
   <xs:enumeration value="Golf"/>
   <xs:enumeration value="BMW"/>
   <xs:enumeration value=""/>
  </xs:restriction>
 </xs:simpleType>
</xs:element>
```

- **Longitud**: Se utiliza en tipos de datos de texto. Define el número exacto de caracteres permitidos, o el mínimo y máximo de ellos.

```
<xs:element name="clave" minOccurs="1" maxOccurs="1">
 <xs:simpleType>
  <xs:restriction base="xs:string">
   <xs:minLength value="5"/>
   <xs:maxLength value="8"/>
  </xs:restriction>
 </xs:simpleType>
</xs:element>
```

- **Plantilla de caracteres**: Se utiliza en tipos de datos de texto. Especifica un patrón o expresión regular que debe cumplir el contenido del elemento. La siguiente tabla muestra algunos de los caracteres que tienen un significado especial para la generación de las máscaras.

Patrón	Significado	Patrón	Significado
[A-Z a-z]	Letra.	AB	Cadena que es la concatenación de las cadenas A y B.
[A-Z]	Letra mayúscula.	A?	Cero o una vez la cadena A.
[a-z]	Letra minúscula.	A+	Una o más veces la cadena A.
[0-9]	Dígitos decimales.	A*	Cero o más veces la cadena A.
\D	Cualquier carácter excepto un dígito decimal.	[abcd]	Alguno de los caracteres que están entre corchetes.
(A)	Cadena que coincide con A.	[^abcd]	Cualquier carácter que no esté entre corchetes.
A | B	Cadena que es igual a la cadena A o a la B.	\t	Tabulación.

EJEMPLO:

```
<xs:element name="iniciales" minOccurs="1" maxOccurs="1">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:pattern value="[A-Z][A-Za-z][A-Za-z]"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
```

- **Tratamiento de espacios en blanco**: Se utiliza en tipos de datos de texto. Especifica cómo se tratan los espacios en blanco (que incluyen también saltos de línea y tabuladores). Los puede respetar (*preserve*), los puede reducir a uno (*collapse*) y los puede reemplazar (*replace*).

```
<xs:element name="direccion" minOccurs="1" maxOccurs="1">
 <xs:simpleType>
  <xs:restriction base="xs:string">
   <xs:whiteSpace value="preserve"/>
  </xs:restriction>
 </xs:simpleType>
</xs:element>
```

También se pueden crear tipos de datos simples basados en **listas de valores** utilizando el atributo derivedBy de simpleType.

Ejercicio resuelto

Creación de una lista con los días de la semana en letras.

Ocultar Información

```
<xs:simpleType name="dia_semana" base="xs:string" derivedBy="list"/>
   <dia_semana>Lunes Martes Miercoles Jueves Viernes Sabado Domingo<dia_semana>
</xs:simpleType>
```

c. También pueden **definirse asignándoles un nombre** y pudiéndose usar en cualquier elemento del documento mediante el atributo *type*. Esta sección es indiferente colocarla antes o después de la definición del elemento raíz:

```
<xs:simpleType name="longitudMaxima">
 <xs:restriction base="xs:string">
  <xs:minLength value="0"/>
  <xs:maxLength value="10"/>
 </xs:restriction>
</xs:simpleType>
<xs:element name="nombre" type="longitudMaxima"/>
```

7. Elementos compuestos <xs:complexType>

1. **Indicadores:** Permiten establecer las características de los elementos que van a utilizarse dentro de otro elemento, por tanto dentro del Contenido complejo.

- **Indicadores de orden**: Asignan el orden de aparición de los elementos descendientes se pueden utilizar las siguientes opciones, tanto individualmente como combinados.
  - **Secuencia** (*xs:sequence*): Define el orden exacto de aparición de los elementos.
  - **Alternativa** (*xs:choice*): Define una serie de elementos entre los cuales sólo se puede elegir uno de ellos.
  - **Todos** (*xs:all*): Define una serie de elementos que pueden aparecer en cualquier orden.

- **Indicadores de ocurrencia**: Asignan cuántas veces puede aparecer dicho elemento.
  - **Mínimo** *(minOccurs)*: Indica el mínimo número de ocurrencias que deben aparecer de ese elemento en el documento XML. No se puede usar si el elemento padre es *<xs:schema>*. Va desde 0 hasta ilimitado (*unbounded*). Por defecto 1.
  - **Máximo** *(maxOccurs)*: Indica el máximo número de ocurrencias que pueden aparecer de ese elemento en el documento XML. No se puede usar si el elemento padre es *<xs:schema>*. Va desde 0 hasta ilimitado (*unbounded*). Por defecto 1.

- **Indicadores de grupo**: Agrupan un conjunto de elementos o de atributos.
  - **Grupo de elementos** (xs:group): Sirve para agrupar un conjunto de declaraciones de elementos relacionados.
  - **Grupo de atributos** (xs:attributeGroup): sirve para agrupar un conjunto de declaraciones de atributos relacionados.

```
<xs:element name="alumno" minOccurs="1" maxOccurs="1">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="nif" type="xs:string" />
      <xs:choice minOccurs="1" maxOccurs="1">
        <xs:element name="nombre" type="xs:string" minOccurs="1" maxOccurs="1" />
        <xs:element name="nombre_compuesto" type="xs:string" minOccurs="1" maxOccurs="1" />
      </xs:choice>
      <xs:element name="apellido" type="xs:string" minOccurs="2" maxOccurs="2" />
      <xs:group ref="direccion" minOccurs="1" maxOccurs="1" />
    </xs:sequence>
```

```
  </xs:complexType>
</xs:element>

<xs:group name="direccion">
  <xs:sequence>
    <xs:element name="calle" type="xs:string"/>
    <xs:element name="localidad" type="xs:string"/>
    <xs:element name="provincia" type="xs:string"/>
  </xs:sequence>
</xs:group>
```

2. **Indicadores:** Permiten establecer las características de los elementos que van a utilizarse dentro de otro elemento, por tanto dentro del Contenido complejo.

2.1. **Elementos vacíos:** Se trata de elementos de tipo de dato complejo y contenido complejo. Estos elementos no podrán contener ninguna clase de contenido. Si nos fijamos no existe el atributo "type".

XSD

```
<xs:element name="h1" minOccurs="1" maxOccurs="1">
  <xs:complexType />
</xs:element>
```

XML

```
<h1></h1>
<h1 />
```

2.2. **Elementos que contienen sólo elementos:** Se trata de elementos de tipo de dato complejo y contenido complejo. Estos elementos definen los elementos que pueden contener en su interior, así como el orden de ellos, su cardinalidad y su agrupamiento

XSD

```
<xs:element name="alumno" minOccurs="1" maxOccurs="1">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="nombre" type="xs:string" minOccurs="1" maxOccurs="1"/>
        <xs:element name="apellido" type="xs:string" minOccurs="2"
maxOccurs="2"/>
      </xs:sequence>
  </xs:complexType>
</xs:element>
```

XML

```
<alumno>
   <nombre>Juan</nombre>
   <apellido>Luque</apellido>
   <apellido>Ostos</apellido>
</alumno>
```

XSD

```
<xs:element name="alumno" minOccurs="1" maxOccurs=" unbounded ">
 <xs:complexType>
  <xs:sequence>
   <xs:choice minOccurs="1" maxOccurs="1">
    <xs:element name="nombre" type="xs:string" minOccurs="1" maxOccurs="1" />
    <xs:element name="nombre_compuesto" type="xs:string" minOccurs="1"
maxOccurs="1" />
   </xs:choice>
   <xs:element name="apellido" type="xs:string" minOccurs="2" maxOccurs="2" />
  </xs:sequence>
 </xs:complexType>
</xs:element>
```

XML

```
<alumno>
   <nombre>Juan</nombre>
   <apellido>Luque</apellido>
   <apellido>Ostos</apellido>
</alumno>
<alumno>
   <nombre_compuesto>Juan Luis</nombre>
   <apellido>Luque</apellido>
   <apellido>Ostos</apellido>
</alumno>
```

2.3. Elementos que contienen datos y atributos: Estos elementos podrán contener sólo datos. Se definen atributos para el elemento.

XSD

```
<xs:element name="producto" minOccurs="1" maxOccurs="1">
 <xs:complexType>
  <xs:simpleContent>
   <xs:extension base="xs:string">
    <xs:attribute name="cod" type="xs:integer" use="optional" />
   </xs:extension>
  </xs:simpleContent>
 </xs:complexType>
</xs:element>
```

XML

```
<producto cod="P01">Manzana</producto>
```

2.4. Elementos que contienen datos y atributos restringidos: Estos elementos podrán contener sólo datos. Se definen atributos para el elemento que contienen restricciones.

XSD

```
<xs:element name="producto" minOccurs="1" maxOccurs="1">
 <xs:complexType>
  <xs:simpleContent>
   <xs:extension base="xs:string">
    <xs:attribute name="cod" type="xs:integer" use="optional" />
      <xs:simpleType>
        <xs:restriction base="xs:string">
          <xs:pattern value="[A-Z][0-9]+" />
        </xs:restriction>
      </xs:simpleType>
   </xs:extension>
  </xs:simpleContent>
 </xs:complexType>
</xs:element>
```

XML

```
<producto cod="P01">Manzana</producto>
```

2.5. Elementos que contienen otros elementos y atributos: Estos elementos definen los elementos que pueden contener en su interior, así como el orden de ellos y su cardinalidad. Además pueden definir atributos.

XSD

```
<xs:element name="alumno" minOccurs="1" maxOccurs="1">
  <xs:complexType>
   <xs:sequence>
    <xs:element name="nombre" type="xs:string" minOccurs="1" maxOccurs="1"/>
    <xs:element name="apellido" type="xs:string" minOccurs="2" maxOccurs="2"/>
   </xs:sequence>
   <xs:attribute name="dni" use="optional">
    <xs:simpleType>
     <xs:restriction base="xs:string">
```

XML

```
< <alumno dni="10203040A">
  <nombre>Juan</nombre>
  <apellido>Luque</apellido>
  <apellido>Ostos</apellido>
</alumno>
```

```
        <xs:pattern value="[0-9]{8}[A-Z]" />
       </xs:restriction>
      </xs:simpleType>
    </xs:attribute>
  </xs:complexType>
</xs:element>
```