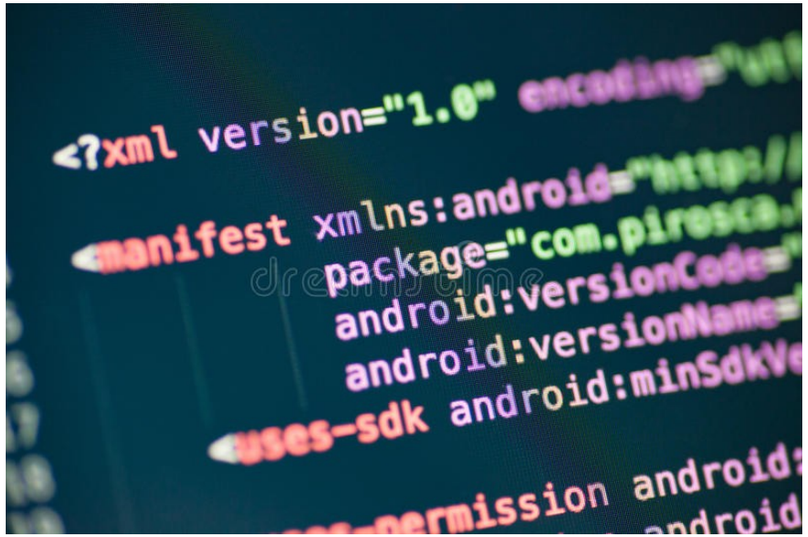
Lenguajes de marcas y sistemas de gestión de la información



UT05 – DTD y XML Schema

3 – XSD

IES Clara del Rey – Madrid

XSD

Xml **S**chema **D**efinition – A veces se usa sólo "XML Schema".

Es un lenguaje que permite definir la estructura de un documento XML, facilitando así su validación. Está basado en XML, es un dialecto XML.

Es una alternativa a DTD, que mejora la definición y validación de documentos XML. Permite especificar, entre otras cosas:

- Elementos y atributos que pueden aparecer en un documento.
- El número y orden de los elementos que hay dentro de otro elemento.
- Tipos de datos para elementos y atributos.
- Valores por defecto para elementos y atributos.

Al ser en sí mismo XML, existe un XSD que permite validar un

XSD

La recomendación del W3C para este estándar está disponible en el W3C:

- XML Schema 1.0
  - XML Schema 1.0 Part 0: Primer
  - XML Schema 1.0 Part 1: Structures
  - XML Schema 1.0 Part 2: Datatypes
- XML Schema 1.1
  - XML Schema 1.1 Part 1: Structures
  - XML Schema 1.1 Part 2: Datatypes

XSD – Declaración

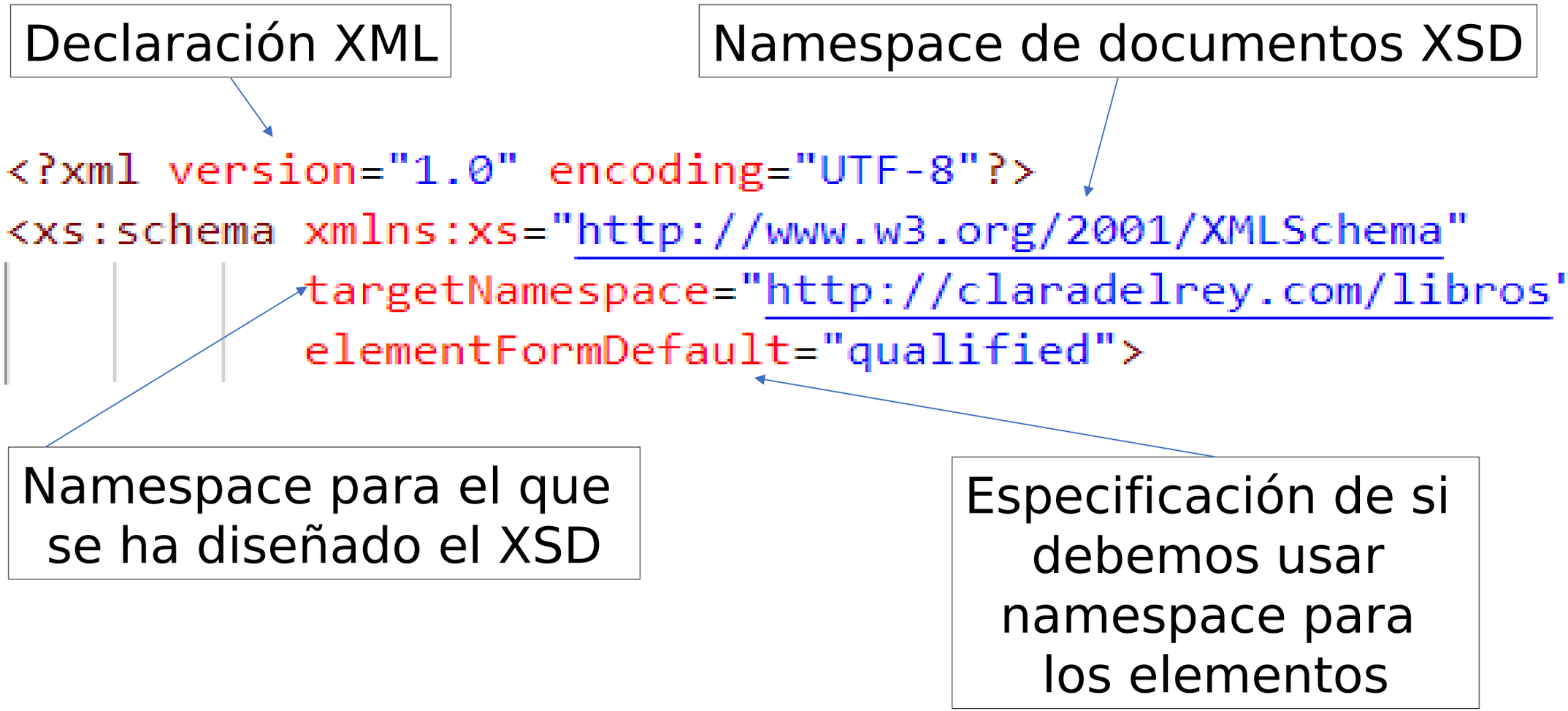
Un esquema XML siempre se define en un documento independiente, no se puede declarar dentro del propio XML, como sí es posible en DTD.

Como XSD es XML, el documento XSD comienza con una declaración XML, y debe tener un elemento raíz, que siempre es "schema", del namespace "http://www.w3.org/2001/XMLSchema".

Como prefijo para este namespace se suele usar "xs" o "xsd".

Al declarar el XSD se puede especificar el namespace para el que se define el esquema, el tipo de XML que queremos validar, y si la validación espera que se usen o no prefijos de namespace en el fichero XML.

XSD - Declaración



XSD – Asociación

En el XML que queremos validar se puede asociar el XSD.

Para hacerlo:

- Declaramos un namespace para XSD, con
- Usando el elemento "schemaLocation" de este namespace, podemos indicar la ubicación de nuestro XSD.
- Si en el XSD se indica un "targetNamespace", este namespace debe aparecer en el documento XML.

XSD – Asociación

Declaración XML

Namespace del documento. Coincide con
el indicado en el XSD en targetNamespace

```
<?xml version="1.0" encoding="UTF-8"?>
<l:libros xmlns:l="http://claradelrey.com/libros"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://claradelrey.com/libros ejemplo01.xsd">
    <l:libro>
        <l:isbn>1215454</l:isbn>
        <l:titulo>Programación en Java </l:titulo>
```

Namespace necesario para poder asociar XSD

Ubicación del XSD. Incluye el namespace del documento XML.

XSD – Declaración y asociación simplificadas

Asumiendo que no vamos a usar namespace en nuestro documento XML, podemos simplificar tanto la declaración como la asociación del XSD.

Dec

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

En la asociación no ponemos namespace, porque no se usará en el XMl

"no

```
<?xml version="1.0" encoding="UTF-8"?>
<libros xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:noNamespaceSchemaLocation="ejemplo02-simple.xsd">
    <libro>
```

XSD – Tipos de datos

Una de las primeras diferencias de XSD respecto a DTD es que se puede especificar el tipo de dato para un atributo o un elemento.

Hay cuatro grandes grupos de tipos de datos

- Cadenas (strings)
- Fechas y horas
- Números
- Otros

Cada uno de estos agrupa varios tipos de datos más específicos.

Además, los tipos de datos pueden personalizarse con restricciones (también llamadas facetas).

XSD – Tipos de datos – Cadenas

xs:string – Admite caracteres, saltos de línea, tabulaciones... El procesador XML no elimina tabuladores o saltos de línea.

xs:normalizedString – Deriva de xs:string, pero en este caso sí se eliminarán los saltos de línea / tabuladores por un único espacio.

xs:token – Deriva de xs:string, pero el procesador elimina tabuladores, saltos de línea, multiples espacios, y espacio al principio / final.

xs:ID – Identificador.

xs:IDREF / xs:IDREFS – Referencia a uno / varios ID del mismo XML.

xs:languaje – Código de idioma.

Otros: ENTITY, ENTITIES, Name, NCName, Qname, NMTOKEN,

XSD – Tipos de datos – Fechas y horas

xs:date – Fecha con formato YYYY-MM-DD.

Se pueden especificar la zona horaria en el dato:

- UTC (GMT): <fecha>2022-02-01Z</fecha>
- Otra zona: <fecha>2022-02-01:00</fecha>
- Otra zona: <fecha>2022-02-01+05:00</fecha>

xs:time – Hora con formato HH:MM:SS.

También se pueden especificar zonas horarias:

- UTC (GMT): <hora>10:15:25Z</hora>
- Otra: <hora>03:02:00-10:00</hora>

XSD – Tipos de datos – Fechas y horas

xs:datetime – Fecha y hora.

Formato YYYY-MM-DDThh:mm:ss.

La T indica dónde comienza la parte de hora.

Se pueden especificar la zona horaria en el dato.

- Ejemplos:
  - <inicio>2022-02-10T15:20:33</inicio>
  - <activado>2022-05-30T15:45:15Z</activado>
  - <caduca>2025-10-19T20:15:20+05:00</caduca>

XSD – Tipos de datos – Fechas y horas

xs:duration – Intervalo de tiempo.

Formato PnYnMnDTnHnMnS, que puede ser negativo, donde:

- P indica que es periodo, T inicio de parte hora.
- nY, nM, nD indican núm. de años, meses, días.
- nH, nM, nS indican número de horas, min, seg.

Ejemplos:

- <activo>P5Y</activo>
- <iniciado>-PT10H30M</iniciado>

XSD – Tipos de datos – Fechas y horas

Partes de una fecha

- **xs:gDay** – Día, con formato DD
- **xs:gMonth** – Mes, con formato MM
- **xs:gMonthDay** – Mes y día, con formato MM-DD
- **xs:gYear** – Año, con formato YYYY
- **xs:gYearMonth** – Año y mes, con formato YYYY-MM

XSD – Tipos de datos – Números

De propósito general:

- **xs:decimal** – Con decimales.
- **xs:integer** – Sin decimales.

Restringiendo el rango:

- **xs:negativeInteger** – Negativos, no incluye el cero (-1, -2, …)
- **xs:postiveInteger** – Positivos, no incluye el cero (1, 2, …)
- **xs:nonNegativeInteger** –Positivos, incluye el cero (0, 1, 2, …)
- **xs:nonPostiveInteger** – Negativos, incluye el cero (0, -1, -2, …)

XSD – Tipos de datos – Números

Otros, para adaptarnos a tamaños de tipos de datos:

- **xs:byte** – Byte (8 bits): de -127 a 127
- **xs:unsignedByte** – Byte (8 bits) sin signo: de 0 a 255
- **xs:int** – Entero de 32 bits con signo
- **xs:unsignedInt** – Entero de 32 bits sin signo
- **xs:long** – Entero de 64 bits con signo
- **xs:unsignedLong** – Entero de 64 bits sin signo
- **xs:short** – Entero de 16 bits con signo
- **xs:unsignedShort** – Entero de 16 bits sin signo

XSD – Tipos de datos – Otros

Valores lógicos (verdadero / falso)

- **xs:boolean** – Valores válidos para un elemento / atributo boolean:
  - true / 1 (que equivale a true)
  - false / 0 (que equivale a false)

Binarios: imágenes, ficheros adjuntos, etc.

- **xs:base64Binary** – Binario codificado usando Base64.
- **xs:hexBinary** – Binario codificado usando código hexadecimal.
- **xs:anyURI** – URI/URL (si hay espacios se deben escapar con %20)

XSD – Elementos simples

Un elemento simple (simpleType) se refiere a un elemento XML que sólo contiene texto. No puede contener otros elementos, ni atributos:

```
<titulo>Desarrollo de interfaces</titulo>
```

En

```
<xs:element name="titulo" type="xs:string" />
```

XSD – Elementos simples
Valores por defecto / valores fijos

Se puede establecer que un elemento debe tener un valor fijo, siempre el mismo.

```
<xs:element name="prestado" type="xs:string" fixed="N" />
```

Se puede establecer valor por defecto, que será el que se utilice si no se indica un valor distinto en el XML.

```
<xs:element name="ejemplares" type="xs:int" default="1" />
```

No se pueden usar los dos a la vez.

Un elemento no puede tener un valor por defecto y otro fijo.

XSD – Atributos

Se declaran casi igual que los elementos:

```
<xs:attribute name="fechaPub" type="xs:date" />
```

También pueden tener valores fijos y por defecto

```
<xs:attribute name="idioma"
              type="xs:string" default="es" />
<xs:attribute name="prestable"
              type="xs:boolean" fixed="true" />
```

XSD – Atributos obligatorios y prohibidos

Atributo obligatorio:

```
<xs:attribute name="autor"
              type="xs:string" use="required" />
```

Atributo prohibido:

```
<xs:attribute name="codigo"
              use="prohibited"/>
```

Por defecto, de forma implícita, los atributos son opcionales.

Se puede indicar de forma explícita que un atributo es opcional con *use=“optional”*.

XSD – Elementos complejos

Un elemento complejo (complexType) es el que contiene otros elementos y/o atributos.

Puede ser una combinación de elementos y atributos, o incluso texto. Así hay múltiples combinaciones.

- Sólo atributos
- Sólo elementos hijos
- Elementos y atributos
- Contenido mixto (texto y elementos hijos)
- Etc.

XSD – Elementos complejos

Por ejemplo, esto sería un elemento complejo:

```
<libro fechaPub="2020-10-17"
       autor="Juan López"
       idioma="es" prestable="true" >
    <isbn>1215454</isbn>
    <titulo>Programación en Java </titulo>
    <prestado>N</prestado>
    <ejemplares>1</ejemplares>
</libro>
```

Este elemento tiene atributos y elementos hijos.

XSD – Elementos complejos

Se usa “complexType” para indicar que es complejo y “sequence” para declarar elementos y atributos.

```
<xs:element name="employee">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="firstname" type="xs:string"/>
      <xs:element name="lastname" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

Si se necesita reutilizar una definición de tipo complejo, para no tener que repetir la misma secuencia en múltiples ocasiones, se puede declarar un tipo personalizado

XSD – Tipos personalizados complejos

Un tipo complejo es el equivalente a una clase en Java. Define una estructura que puede reutilizarse. Se definen con complexType, asignando un nombre al tipo:

```
<xs:complexType name="personinfo">
  <xs:sequence>
    <xs:element name="firstname" type="xs:string"/>
    <xs:element name="lastname" type="xs:string"/>
  </xs:sequence>
</xs:complexType>
```

Que después se utiliza al definir un elemento.

```
<xs:element name="employee" type="personinfo"/>
```

XSD – Tipos personalizados complejos

Esto permite reutilizar el tipo en múltiples elementos:

```
<xs:element name="employee" type="personinfo"/>
<xs:element name="student" type="personinfo"/>
<xs:element name="member" type="personinfo"/>


<xs:complexType name="personinfo">
  <xs:sequence>
    <xs:element name="firstname" type="xs:string"/>
    <xs:element name="lastname" type="xs:string"/>
  </xs:sequence>
</xs:complexType>
```

XSD – Extensión de tipos personalizados

Es posible definir un tipo a partir de la definición de otro, funciona de forma similar a la herencia en Java

```
<xs:complexType name="personinfo">
  <xs:sequence>
    <xs:element name="firstna
    <xs:element name="lastnam
  </xs:sequence>
</xs:complexType>
```

```
<xs:complexType name="fullpersoninfo">
  <xs:complexContent>
    <xs:extension base="personinfo">
      <xs:sequence>
        <xs:element name="address" type="xs:string"/>
        <xs:element name="city" type="xs:string"/>
        <xs:element name="country" type="xs:string"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

XSD – Ejemplos de tipos complejos

Elemento complejo vacío (sólo atributos):

```
<product prodid="1345" />
```

```
<xs:element name="product">
  <xs:complexType>
    <xs:attribute name="prodid" type="xs:positiveInteger"/>
  </xs:complexType>
</xs:element>
```

```
<xs:element name="product" type="prodtype"/>

<xs:complexType name="prodtype">
  <xs:attribute name="prodid" type="xs:positiveInteger"/>
</xs:complexType>
```

XSD – Ejemplos de tipos complejos

Elemento que sólo contiene otros elementos:

```
<person>
  <firstname>John</firstname>
  <lastname>Smith</lastname>
</person>
```

```
<xs:element name="person" type="persontype"/>

<xs:complexType name="persontype">
  <xs:sequence>
    <xs:element name="firstname" type="xs:string"/>
    <xs:element name="lastname" type="xs:string"/>
  </xs:sequence>
</xs:complexType>
```

XSD – Indicadores

Establecen cómo se deben usar los elementos en el XML. Hay siete:

- De orden:
  - secuencia (sequence),
  - todo (all)
  - elección (choice)
- De ocurrencia (cantidad):
  - maxOccurs y minOccurs.
- De grupo:
  - grupo de elementos (group)
  - grupo de atributos (attributeGroup)

XSD – Indicadores de orden

xs:sequence – Los elementos deben aparecer obligatoriamente en el orden establecido.

```
<xs:element name="ciudad">
    <xs:complexType>
        <xs:sequence>
            <xs:element name="pais" type="xs:string" />
            <xs:element name="nombre" type="xs:string" />
        </xs:sequence>
    </xs:complexType>
</xs:element>
```

```
<ciudad>
    <pais>Italia</pais>
    <nombre>Florencia</nombre>
</ciudad>
<ciudad>
    <nombre>Florencia</nombre>
    <pais>Italia</pais>
</ciudad>
```

XSD – Indicadores de orden

xs:all – Los elementos pueden aparecer en cualquier orden.

```
<xs:element name="ciudad">
    <xs:complexType>
        <xs:all>
            <xs:element name="pais" type="xs:string" />
            <xs:element name="nombre" type="xs:string" />
        </xs:all>
    </xs:complexType>
</xs:element>
```

```
<ciudad>
    <pais>Italia</pais>
    <nombre>Florencia</nombre>
</ciudad>
<ciudad>
    <nombre>Florencia</nombre>
    <pais>Italia</pais>
</ciudad>
```

XSD – Indicadores de orden

xs:choice – Solo se puede utilizar uno de los elementos hijo.

```
<xs:element name="ciudad">
    <xs:complexType>
        <xs:choice>
            <xs:element name="pais" type="xs:string" />
            <xs:element name="nombre"
        </xs:choice>
    </xs:complexType>
</xs:element>
```

```
<ciudad>
    <pais>Italia</pais>
</ciudad>
<ciudad>
    <nombre>Florencia</nombre>
</ciudad>    <ciudad>
    <nombre>Florencia</nombre>
    <pais>Italia</pais>
</ciudad>
```

XSD – Indicadores de ocurrencia (cantidad)

xs:minOccurs y xs:maxOccurs – Número mínimo y máximo de veces que que debe aparecer el elemento.

```
<xs:element name="ciudad" maxOccurs="2">
    <xs:complexType>
        <xs:sequence>
            <xs:element name="pais" type="xs:st
            <xs:element name="nombre" type="xs
        </xs:sequence>
    </xs:complexType>
</xs:element>
```

```
<ciudad>
    <pais>Italia</pais>
    <nombre>Florencia</nombre>
</ciudad>
<ciudad>
    <pais>Italia</pais>
    <nombre>Florencia</nombre>
</ciudad>
<ciudad>
    <pais>Italia</pais>
    <nombre>Florencia</nombre>
</ciudad>
```

XSD – Indicadores de ocurrencia (cantidad)

El valor por defecto para minOccurs y maxOccurs es 1.

Si se usa “0” en minOccurs el elemento será opcional.

Si se usa “unbounded” en maxOccurs, es que no hay límite.

```
<xs:element name="pais" type="xs:string" />
<xs:element name="nombre" type="xs:string" />
<xs:element name="restaurante" type="xs:string" minOccurs="0" maxOccurs="unbounded"></xs:element>
```

```
<ciudad>
    <pais>Italia</pais>
    <nombre>Florencia</nombre>
</ciudad>
<ciudad>
    <pais>España</pais>
    <nombre>Madrid</nombre>
    <restaurante>El Llar</restaurante>
    <restaurante>La Playa</restaurante>
    <restaurante>De Manila</restaurante>
</ciudad>
```

XSD – Indicadores de grupo – Elementos

Permite definir un grupo de elementos, que se puede referenciar en otra parte del XSD.

```
<xs:group name="persongroup">
  <xs:sequence>
    <xs:element name="firstname" type="xs:string"/>
    <xs:element name="lastname" type="xs:string"/>
    <xs:element name="birthday" type="xs:date"/>
  </xs:sequence>
</xs:group>
```

```
<xs:complexType name="personinfo">
  <xs:sequence>
    <xs:group ref="persongroup"/>
    <xs:element name="country" type="xs:string"/>
  </xs:sequence>
pe>
```

```
<xs:element name="person" type="personinfo"/>
```

XSD – Indicadores de grupo – Atributos

Permite definir un grupo de atributos, que se puede referenciar en otra parte del XSD.

```
<xs:attributeGroup name="personattrgroup">
  <xs:attribute name="firstname" type="xs:string"/>
  <xs:attribute name="lastname" type="xs:string"/>
  <xs:attribute name="birthday" type="xs:date"/>
</xs:attributeGr
```

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

XSD – any y anyAttribute

Permiten incluir en el XML elementos no especificados en el esquema, que suelen ser de otro namespace, por lo que hay que usar la forma “compleja” de referenciar los XSD.

```
<xs:element name="person">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="firstname" type="xs:string"/>
      <xs:element name="lastna
      <xs:any minOccurs="0"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

```
<xs:complexType>
  <xs:sequence>
    <xs:element name="firstname" type="xs:string"/>
    <xs:element name="lastname" type="xs:string"/>
  </xs:sequence>
  <xs:anyAttribute/>
</xs:complexType>
```

XSD – Restricciones a tipos de datos

Las restricciones, también denominadas "facetas", permiten adaptar los tipos de datos a las necesidades específicas de nuestro esquema.

Limitan los valores admitidos para elementos o atributos.

Se establece un tipo de datos "inicial", al que se añaden ciertos criterios para concretar las características de los tipos "válidos".

Ejemplo: que un atributo debe ser de tipo entero positivo, pero, además, que debe tener exactamente ocho dígitos.

XSD – Restricciones a tipos de datos

Restricción	Efecto
minExclusive	Mínimo. No incluido. Valor > restricción.
minInclusive	Mínimo. Incluido. Valor >= restricción.
maxExclusive	Máximo. No incluido. Valor < restricción.
maxInclusive	Máximo. Incluido. Valor <= restricción.
maxLength	Longitud máxima. >= 0
minLength	Longitud mínima. >= 0.
Enumeration	Define lista de valores aceptados
totalDigits	Número exacto de dígitos. >= 0.
fractionDigits	Número máximo de decimales. >= 0.
length	Número máximo (caracteres/elementos). >=0.
pattern	Expresión regular.
whiteSpace	Preserve, replace o collapse

XSD – Restricciones – Ejemplos

```
<xs:element name="age">
  <xs:simpleType>
    <xs:restriction base="xs:integer">
      <xs:minInclusive value="0"/>
      <xs:maxInclusive value="120"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
```

```
<xs:element name="car">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:enumeration value="Audi"/>
      <xs:enumeration value="Golf"/>
      <xs:enumeration value="BMW"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
```

XSD – Restricciones – Ejemplos

```
<xs:element name="letter">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:pattern value="[a-z]"/>
    </xs:restriction>
  </xs:simpleTyp
</xs:element>
```

```
<xs:element name="initials">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:pattern value="[a-zA-Z][a-zA-Z][a-zA-Z]"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
```

XSD – Restricciones – Ejemplos

```
<xs:element name="password">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:length value="8"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
```

```
<xs:element name="password">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:minLength value="5"/>
      <xs:maxLength value="8"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
```