Tema 3: XML Schema Definition (XSD)

1. Introducción

XML Schema Definition es una nueva especificación para definir vocabularios XML, más moderna que DTD y más completa.

Las características más importantes que aporta XSD son:

- Está escrito en XML y, por lo tanto, no hay que aprender un lenguaje nuevo para definir esquemas XML.
- Tiene su propio sistema de datos, de forma que se podrá comprobar el contenido de los elementos.
- Soporta espacios de nombres para permitir mezclar diferentes vocabularios.
- Permite ser reutilizado y sigue los modelos de programación como herencia de objetos y sustitución de tipos.

1.1 Enlazar un documento XML con un documento XSD

Partimos de un documento XML cuya etiqueta raíz es *receta*:

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

<receta>
      ...
</receta>
```

Lo que hacemos es añadir en esa etiqueta raíz una declaración de espacio de nombres:

```
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
```

Y enlazamos con el documento XSD con el atributo mediante el atributo:

```
xsi:noNamespaceSchemaLocation="receta.xsd"
```

Quedando el documento así:

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

<receta xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:noNamespaceSchemaLocation="receta.xsd">
       ...
</receta>
```

1.2 Estructura de un documento XSD

El documento XSD es un también un documento XML, luego tiene que cumplir con todas las reglas de sintaxis de XML. Su etiqueta raíz es siempre *xs:schema*, e incluye una declaración de espacio de nombres :

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

<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">

       ...

</xs:schema>
```

Para cada elemento del documento XML aparecerá un elemento <*xs:element*> en el XSD.

2. Elementos con contenido de tipo simple

Son elementos con contenido simple aquéllos que no contienen atributos ni otros elementos. Para indicar el tipo de contenido disponemos muchas opciones, algunas de las cuales son:

xs:string	Cadena de texto
xs:decimal	Número decimal
xs:integer	Número entero
xs:positiveInteger	Entero mayor que cero
xs:nonNegativeInteger	Entero mayor o igual que cero
xs:date	Fecha (con formato aaaa-mm-dd)
xs:time	Hora (con formato hh:mm)

receta.xml

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

<receta xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:noNamespaceSchemaLocation="receta.xsd">
       Paracetamol
</receta>
```

receta.xsd

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

<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">

       <xs:element name="receta" type="xs:string" />

</xs:schema>
```

2.1 Cardinalidad de los elementos (minOccurs, maxOccurs)

Por defecto si no indicamos cardinalidad en un elemento, éste tiene que aparecer exactamente una vez. Para indicar otra cantidad usamos *minOccurs* y *maxOccurs* (sus valores por defecto son 1).

Algunos casos particulares:

- minOccurs = "0": el elemento puede no aparecer.
- maxOccurs = "unbounded": el elemento puede aparecer cuantas veces se quiera.

En el siguiente ejemplo, la etiqueta *medicamento* puede aparacer hasta 10 veces (también puede no aparecer):

receta.xml

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

	<medicamento>Omeprazol</medicamento>
	<medicamento>Paracetamol</medicamento>

</receta>
```

receta.xsd

```
<?xml version="1.0" encoding="UTF-8" ?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
	<xs:element name="receta">
		<xs:complexType>
			<xs:sequence>
				<xs:element name="medicamento" minOccurs="0"
					maxOccurs="10" />
			</xs:sequence>
		</xs:complexType>
	</xs:element>
</xs:schema>
```

2.2 Atributo fixed

El atributo *fixed* sirve para indicar que un elemento sólo puede tener el valor indicado. En el siguiente ejemplo, el elemento f*armacia* sólo puede tener el valor "Farmacia Juaneda". También podría aparecer vacío, porque se entiende que entonces tiene ese mismo valor.

receta.xml

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

<receta xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
	xsi:noNamespaceSchemaLocation="receta.xsd">

	<medicamento>Omeprazol</medicamento>
	<medicamento>Paracetamol</medicamento>

	<farmacia>Farmacia Juaneda</farmacia>

</receta>
```

receta.xsd

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

<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">

	<xs:element name="receta">
		<xs:complexType>
			<xs:sequence>

				<xs:element name="medicamento" minOccurs="0"
					maxOccurs="10" />
				<xs:element name="farmacia" fixed="Farmacia Juaneda" />

			</xs:sequence>
		</xs:complexType>
	</xs:element>

</xs:schema>
```

2.3 Atributo default

El atributo *default* permite indicar un valor por defecto (en caso de que el elemento estuviera vacío). A diferencia de *fixed*, este valor puede cambiarse en el XML.

```
	<xs:element name="farmacia" type="xs:string" default="Farmacia Juaneda"/>
```

3. Restricciones de tipos simples personalizados

Podemos poner restricciones a cualquier tipo simple. Para ello escribimos un bloque así:

```
<xs:element name="edad">
	<xs:simpleType>
		<xs:restriction base="tipo">
			(restricciones que queramos añadir)
		</xs:restriction>
	</xs:simpleType>
</xs:element>
```

3.1 Restricciones: minInclusive, minExclusive, maxInclusive, maxExclusive

Sirven para poner valores mínimos (minInclusive o minExclusive) o máximos (maxInclusive, maxExclusive) a un elemento o atributo. "Inclusive" significa que el valor indicado se incluye, y "exclusive" lo contrario.

En el siguiente ejemplo indicamos que la edad debe estar entre 0 y 150 (ambos inclusive):

receta.xml

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

<receta xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
	xsi:noNamespaceSchemaLocation="receta.xsd">

	<medicamento>Omeprazol</medicamento>
	<medicamento>Paracetamol</medicamento>

	<farmacia>Farmacia Juaneda</farmacia>

	<edad>70</edad>

</receta>
```

receta.xsd

```
<xs:element name="edad">
	<xs:simpleType>
		<xs:restriction base="xs:integer">
			<xs:minInclusive value="0" />
			<xs:maxInclusive value="150" />
		</xs:restriction>
	</xs:simpleType>
</xs:element>
```

3.2 Restricciones: minLength, maxLength, length

Los elementos *minLength* y *maxLength* permiten imponer una longitud mínima o máxima a un cadena de texto. El elemento *length* indica una longitud total.

En el siguiente ejemplo indicamos que el nombre del paciente no puede superar los cincuenta caracteres:

receta.xml

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

<receta xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
	xsi:noNamespaceSchemaLocation="receta.xsd">

	<medicamento>Omeprazol</medicamento>
	<medicamento>Paracetamol</medicamento>

	<farmacia>Farmacia Juaneda</farmacia>

	<edad>70</edad>
	<nombre>Francisco Santos</nombre>

</receta>
```

receta.xsd

```
<xs:element name="nombre">
	<xs:simpleType>
		<xs:restriction base="xs:string">
			<xs:maxLength value="50" />
		</xs:restriction>
	</xs:simpleType>
</xs:element>
```

3.3 Restricción: enumeration

Con el elemento *enumeration* podemos limitar los posibles valores de un elemento o atributo a una lista.

En el siguiente ejemplo, el único contenido de la etiqueta *pago* puede ser "Íntegro", "Copago" o "Gratuito":

receta.xml

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

<receta xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
	xsi:noNamespaceSchemaLocation="receta.xsd">

	<medicamento>Omeprazol</medicamento>
	<medicamento>Paracetamol</medicamento>

	<farmacia>Farmacia Juaneda</farmacia>

	<edad>70</edad>
	<nombre>Francisco Santos</nombre>

	<pago>Gratuito</pago>

</receta>
```

receta.xsd

```
<xs:element name="pago">
	<xs:simpleType>
		<xs:restriction base="xs:string">
			<xs:enumeration value="Íntegro" />
			<xs:enumeration value="Copago" />
			<xs:enumeration value="Gratuito" />
		</xs:restriction>
	</xs:simpleType>
</xs:element>
```

3.4 Restricción: pattern

Podemos usar patrones que indicarán cómo son los valores válidos para un elemento o atributo. La lista de caracteres que podemos usar es:

Símbolo	**Equivalencia**
.	Cualquier carácter
\d	Cualquier dígito
\D	Cualquier carácter no dígito
x*	Lo de delante de * tiene que aparecer 0 o más veces
x+	Lo de delante de + tiene que aparecer 1 o más veces
x?	Lo de delante de ? puede aparecer o no
[abc]	Tiene que haber algún carácter de los de dentro
[0-9]	Tiene que haber un valor entre los dos especificados, ambos inclusive
x{5}	Tiene que aparecer 5 veces lo de delante
x{5,}	Tiene que aparecer 5 o más veces lo de delante
x{5,8}	Tiene que aparecer entre 5 y 8 veces lo de delante
(valor1|valor2|valor3)	Uno de los valores

En el siguiente ejemplo indicamos que un DNI tiene que estar formado por 8 cifras y una letra mayúscula:

receta.xml

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

<receta xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
	xsi:noNamespaceSchemaLocation="receta.xsd">

	<medicamento>Omeprazol</medicamento>
	<medicamento>Paracetamol</medicamento>

	<farmacia>Farmacia Juaneda</farmacia>

	<edad>70</edad>
	<nombre>Francisco Santos</nombre>

	<pago>Gratuito</pago>

	<dni>22199029H</dni>

</receta>
```

receta.xsd

```
<xs:element name="dni">
	<xs:simpleType>
		<xs:restriction base="xs:string">
			<xs:pattern value="\d{8}[A-Z]" />
		</xs:restriction>
	</xs:simpleType>
</xs:element>
```

3.5 Restricción: totalDigits y fractionDigits

En caso de que tratemos con valores decimales, podemos usar dos restricciones más:

- totalDigits: El número de dígitos debe ser menor o igual al indicado.
- fractionDigits: El número de cifras decimales debe ser menor o igual al indicado.

Usaremos estas restricciones en algunos ejercicios más adelante.

4. Elementos de tipo complejo

4.1 Elementos con contenido textual y atributos

Un elemento con contenido textual y atributos se expresa como una extensión de un contenido simple añadiéndole un atributo.

Podemos indicar que el atributo es opcional u obligatorio con las siguientes instrucciones:

use="optional"
use="required"

Si no aparece "use", se entiende que el atributo es opcional.

receta.xml

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

<receta xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
	xsi:noNamespaceSchemaLocation="receta.xsd">

	<medicamento>Omeprazol</medicamento>
	<medicamento>Paracetamol</medicamento>

	<farmacia>Farmacia Juaneda</farmacia>

	<edad>70</edad>
	<nombre>Francisco Santos</nombre>

	<pago>Gratuito</pago>

	<dni>22199029H</dni>

	<medico numColegiado="171612713">Juan Ríos</medico>

</receta>
```

receta.xsd

```
<xs:element name="medico">
	<xs:complexType>
		<xs:simpleContent>
			<xs:extension base="xs:string">
				<xs:attribute name="numColegiado" type="xs:integer"
					use="required" />
			</xs:extension>
		</xs:simpleContent>
	</xs:complexType>
</xs:element>
```

4.2 Elementos que contienen otros elementos

Los elementos hijos deben ir dentro de <xs:sequence>, <xs:choice> o <xs:all>.

- xs:sequence: Los elementos van apareciendo en el orden indicado.
- xs:choice: Aparecerá un elemento de los que hay en la lista.
- xs:all: Los elementos pueden aparecer en cualquier orden.

En el ejemplo se indica que dentro de *posología* aparecerá una secuencia con *tomas* primero y *horario* después, en ese orden:

receta.xml

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

<receta xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
	xsi:noNamespaceSchemaLocation="receta.xsd">

	<medicamento>Omeprazol</medicamento>
	<medicamento>Paracetamol</medicamento>

	<farmacia>Farmacia Juaneda</farmacia>

	<edad>70</edad>

	<nombre>Francisco Santos</nombre>

	<pago>Gratuito</pago>

	<dni>22199029H</dni>

	<medico numColegiado="171612713">Juan Ríos</medico>

	<posologia>
		<tomas>3</tomas>
		<horario>Mañana, tarde y noche</horario>
	</posologia>

</receta>
```

receta.xsd

```
<xs:element name="posologia">
	<xs:complexType>
		<xs:sequence>
			<xs:element name="tomas" type="xs:positiveInteger" />
			<xs:element name="horario" type="xs:string" />
		</xs:sequence>
	</xs:complexType>
</xs:element>
```

4.3 Elementos que contienen atributos y otros elementos

Si un elemento contiene otros elementos y, además, atributos, éstos se colocan después de la secuencia de elementos.

En el siguiente ejemplo vemos que *posología* tiene un atributo obligatorio *duración*, además de una secuencia de elementos *tomas* y *horario*:

receta.xml

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

<receta xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
	xsi:noNamespaceSchemaLocation="receta.xsd">

	<medicamento>Omeprazol</medicamento>
	<medicamento>Paracetamol</medicamento>

	<farmacia>Farmacia Juaneda</farmacia>

	<edad>70</edad>

	<nombre>Francisco Santos</nombre>

	<pago>Gratuito</pago>

	<dni>22199029H</dni>

	<medico numColegiado="171612713">Juan Ríos</medico>

	<posologia duracion="Cinco días">
		<tomas>3</tomas>
		<horario>Mañana, tarde y noche</horario>
	</posologia>

</receta>
```

receta.xsd

```
<xs:element name="posologia">
	<xs:complexType>
		<xs:sequence>
			<xs:element name="tomas" type="xs:positiveInteger"/>
			<xs:element name="horario" type="xs:string"/>
		</xs:sequence>
		<xs:attribute name="duracion" type="xs:string" use="required"/>
	</xs:complexType>
</xs:element>
```

4.4 Elementos vacíos

Siempre son de tipo complejo.

En el siguiente ejemplo, el elemento *alergias* indica que el paciente sufre algún tipo de alergia a medicamentos. No debe tener contenido:

receta.xml

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

<receta xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
	xsi:noNamespaceSchemaLocation="receta.xsd">

	<medicamento>Omeprazol</medicamento>
	<medicamento>Paracetamol</medicamento>

	<farmacia>Farmacia Juaneda</farmacia>

	<edad>70</edad>

	<nombre>Francisco Santos</nombre>

	<pago>Gratuito</pago>

	<dni>22199029H</dni>

	<medico numColegiado="171612713">Juan Ríos</medico>

	<posologia>
		<tomas>3</tomas>
		<horario>Mañana, tarde y noche</horario>
	</posologia>

	<alergias/>

</receta>
```

receta.xsd

```
<xs:element name="alergias">
	<xs:complexType />
</xs:element>
```

4.5 Elementos sin contenido pero con atributos

En este caso definimos el elemento como un complexType que contiene el atributo.

receta.xml

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

<receta xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
	xsi:noNamespaceSchemaLocation="receta.xsd">

	<medicamento>Omeprazol</medicamento>
	<medicamento>Paracetamol</medicamento>

	<farmacia>Farmacia Juaneda</farmacia>

	<edad>70</edad>

	<nombre>Francisco Santos</nombre>

	<pago>Gratuito</pago>

	<dni>22199029H</dni>

	<medico numColegiado="171612713">Juan Ríos</medico>

	<posologia>
		<tomas>3</tomas>
		<horario>Mañana, tarde y noche</horario>
	</posologia>

	<alergias />

	<grupo_sanguineo tipo="A+" />

</receta>
```

receta.xsd

```
<xs:element name="grupo_sanguineo">
	<xs:complexType>
		<xs:attribute name="tipo" type="xs:string" />
	</xs:complexType>
</xs:element>
```

4.6 Contenido mixto

Se añade el atributo mixed="true" en el elemento <complexType>. Los elementos hijos van en una elección o secuencia.

En este ejemplo añadimos un bloque *<instrucciones>* con explicaciones del médico. En medio de las explicaciones el doctor puede resaltar algunas partes rodeándolas con etiquetas *<b>*:

receta.xml

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

<receta xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
	xsi:noNamespaceSchemaLocation="receta.xsd">

	<medicamento>Omeprazol</medicamento>
	<medicamento>Paracetamol</medicamento>

	<farmacia>Farmacia Juaneda</farmacia>

	<edad>70</edad>

	<nombre>Francisco Santos</nombre>

	<pago>Gratuito</pago>

	<dni>22199029H</dni>

	<medico numColegiado="171612713">Juan Ríos</medico>

	<posologia>
		<tomas>3</tomas>
		<horario>Mañana, tarde y noche</horario>
	</posologia>

	<alergias />

	<grupo_sanguineo tipo="A+" />

	<instrucciones>
		Esta medicación debe tomarse en <b>ayunas</b>. No deben superarse los
<b>cuatro</b> comprimidos al día.
	</instrucciones>

</receta>
```

receta.xsd

```
<xs:element name="instrucciones">
	<xs:complexType mixed="true">
		<xs:choice maxOccurs="unbounded">
			<xs:element name="b" type="xs:string" />
		</xs:choice>
	</xs:complexType>
```

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

5. Tipos definidos por el usuario

XSD nos permite crear nuestros propios tipos de datos. Por ejemplo, podemos definir un tipo para almacenar edades:

```
<xs:simpleType name="TipoEdad">
	<xs:restriction base="xs:integer">
		<xs:minInclusive value="0" />
		<xs:maxInclusive value="150" />
	</xs:restriction>
</xs:simpleType>
```

Y utilizarlo en nuestro esquema igual que si fuera un tipo predefinido:

```
<xs:element name="edad" type="TipoEdad" />
```

En el siguiente ejemplo podemos ver el XSD anterior tras definir varios tipos de datos personalizados:

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

<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">

	<xs:element name="receta">
		<xs:complexType>
			<xs:sequence>

				<xs:element name="medicamento" minOccurs="0"
					maxOccurs="10" />

				<xs:element name="farmacia" default="Farmacia Juaneda" />

				<xs:element name="edad" type="TipoEdad" />

				<xs:element name="nombre" type="TipoNombre" />

				<xs:element name="pago" type="TipoPago" />

				<xs:element name="dni" type="TipoDni" />

				<xs:element name="medico" type="TipoMedico" />

				<xs:element name="posologia" type="TipoPosologia" />

				<xs:element name="alergias">
					<xs:complexType />
				</xs:element>

				<xs:element name="grupo_sanguineo"
					type="TipoGrupoSanguineo" />

				<xs:element name="instrucciones" type="TipoInstrucciones" />

			</xs:sequence>
		</xs:complexType>
```

```
	</xs:element>

	<!-- Tipos definidos por el usuario -->

	<xs:simpleType name="TipoEdad">
		<xs:restriction base="xs:integer">
			<xs:minInclusive value="0" />
			<xs:maxInclusive value="150" />
		</xs:restriction>
	</xs:simpleType>

	<xs:simpleType name="TipoNombre">
		<xs:restriction base="xs:string">
			<xs:maxLength value="50" />
		</xs:restriction>
	</xs:simpleType>

	<xs:simpleType name="TipoPago">
		<xs:restriction base="xs:string">
			<xs:enumeration value="Gratuito" />
			<xs:enumeration value="Copago" />
			<xs:enumeration value="Íntegro" />
		</xs:restriction>
	</xs:simpleType>

	<xs:simpleType name="TipoDni">
		<xs:restriction base="xs:string">
			<xs:pattern value="\d{8}[A-Z]" />
		</xs:restriction>
	</xs:simpleType>

	<xs:complexType name="TipoMedico">
		<xs:simpleContent>
			<xs:extension base="xs:string">
				<xs:attribute name="numColegiado" type="xs:integer" />
			</xs:extension>
		</xs:simpleContent>
	</xs:complexType>

	<xs:complexType name="TipoPosologia">
		<xs:sequence>
			<xs:element name="tomas" type="xs:positiveInteger" />
			<xs:element name="horario" type="xs:string" />
		</xs:sequence>
	</xs:complexType>

	<xs:complexType name="TipoGrupoSanguineo">
		<xs:attribute name="tipo" type="xs:string" />
	</xs:complexType>

	<xs:complexType name="TipoInstrucciones" mixed="true">
		<xs:choice maxOccurs="unbounded">
			<xs:element name="b" type="xs:string" />
		</xs:choice>
	</xs:complexType>

</xs:schema>
```

6. Esquema de la receta (sin seguir el modelo de muñecas rusas)

Cuando escribimos un XSD complejo acabamos teniendo un XML con muchas indentaciones y elementos dentro de elementos, dentro de elementos... Esto se conoce como modelo de muñecas rusas, por las conocidas *Matrioshkas*, muñecas huecas que guardan dentro otras muñecas.

Para evitar esta complejidad podemos usar el atributo *ref* tanto de *xs:element* como de *xs:attribute*, en vez del atributo *name*. El *ref* nos permite indicar el contenido de un elemento, dejando para después la descripción de dicho contenido.

Éste es el resultado final de la práctica desarrollada a lo largo del tema, evitando usar el modelo de muñecas rusas:

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

<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">

	<xs:element name="receta">
		<xs:complexType>
			<xs:sequence>
				<xs:element ref="medicamento" maxOccurs="unbounded" />
				<xs:element ref="farmacia" />
				<xs:element ref="edad" />
				<xs:element ref="nombre" />
				<xs:element ref="pago" />
				<xs:element ref="dni" />
				<xs:element ref="medico" />
				<xs:element ref="posologia" />
				<xs:element ref="alergias" />
				<xs:element ref="grupo_sanguineo" />
				<xs:element ref="instrucciones" />
			</xs:sequence>
		</xs:complexType>
	</xs:element>

	<xs:element name="medicamento" type="xs:string" />

	<xs:element name="farmacia" type="xs:string" default="Farmacia Juaneda" />

	<xs:element name="edad" type="xs:nonNegativeInteger" />

	<xs:element name="nombre" type="xs:string" />

	<xs:element name="pago">
		<xs:simpleType>
			<xs:restriction base="xs:string">
				<xs:enumeration value="Gratuito" />
				<xs:enumeration value="Copago" />
				<xs:enumeration value="Íntegro" />
			</xs:restriction>
		</xs:simpleType>
	</xs:element>

	<xs:element name="dni">
		<xs:simpleType>
```

```
            <xs:restriction base="xs:string">
                  <xs:pattern value="\d{8}[A-Z]" />
            </xs:restriction>
      </xs:simpleType>
</xs:element>

<xs:element name="medico">
      <xs:complexType>
            <xs:simpleContent>
                  <xs:extension base="xs:string">
                        <xs:attribute ref="numColegiado" />
                  </xs:extension>
            </xs:simpleContent>
      </xs:complexType>
</xs:element>

<xs:attribute name="numColegiado"
      type="xs:positiveInteger" />

<xs:element name="posologia">
      <xs:complexType>
            <xs:sequence>
                  <xs:element ref="tomas" />
                  <xs:element ref="horario" />
            </xs:sequence>
      </xs:complexType>
</xs:element>

<xs:element name="tomas" type="xs:positiveInteger" />

<xs:element name="horario" type="xs:string" />

<xs:element name="alergias">
      <xs:complexType />
</xs:element>

<xs:element name="grupo_sanguineo">
      <xs:complexType>
            <xs:attribute ref="tipo" />
      </xs:complexType>
</xs:element>

<xs:attribute name="tipo">
      <xs:simpleType>
            <xs:restriction base="xs:string">
                  <xs:enumeration value="A" />
                  <xs:enumeration value="A+" />
                  <xs:enumeration value="B" />
                  <xs:enumeration value="B+" />
                  <xs:enumeration value="0" />
                  <xs:enumeration value="0+" />
                  <xs:enumeration value="AB" />
                  <xs:enumeration value="AB+" />
            </xs:restriction>
      </xs:simpleType>
</xs:attribute>

<xs:element name="instrucciones">
      <xs:complexType mixed="true">
```

```
                    <xs:sequence>
                          <xs:element ref="b" maxOccurs="unbounded" />
                    </xs:sequence>
              </xs:complexType>
       </xs:element>

       <xs:element name="b" type="xs:string" />

</xs:schema>
```

7. Actividad final de repaso

En un instituto, al final de cada evaluación, cada profesor envía a secretaría un documento XML con las calificaciones de cada uno de sus módulos.

En el siguiente XML podemos ver las notas de la asignatura "Programación básica", del profesor Marcel Puig:

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

<classe modul="3" nommodul="Programació bàsica"
	xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
	xsi:noNamespaceSchemaLocation="classe.xsd">

	<curs especialitat="ASIX" numero="1" />

	<professor>
		<nom>Marcel</nom>
		<cognom>Puig</cognom>
	</professor>

	<alumnes>
		<alumne>
			<nom>Filomeno</nom>
			<cognom>Garcia</cognom>
			<nota>5</nota>
		</alumne>

		<alumne delegat="sí">
			<nom>Frederic</nom>
			<cognom>Pi</cognom>
			<nota>3</nota>

		</alumne>

		<alumne>
			<nom>Manel</nom>
			<cognom>Puigdevall</cognom>
			<nota>8</nota>
		</alumne>
	</alumnes>

</classe>
```

Las restricciones que se nos dan son las siguientes:

- El atributo *mòdul* debe tomar valores enteros positivos.
- El atributo *especialitat* sólo puede tomar uno de estos valores: DAW, ASIX, DAM.
- El atributo *número* debe ser un entero positivo.
- Debe haber al menos un *alumno* en la clase.
- Una *nota* debe ser un entero entre 0 y 10, ambos inclusive.
- Todos los atributos son obligatorios.