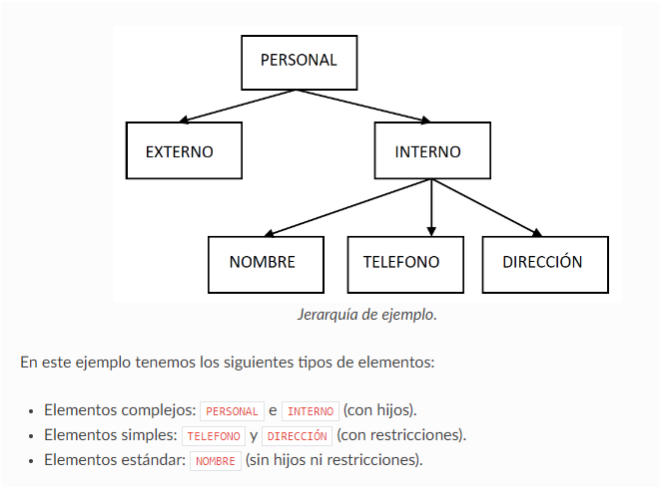
Nodo raíz -schema - prologo

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

Declaracion de elementos

Tipos de elementos

1. elementos estándar
   sin hijos, sin restricciones, sin atributos
2. elementos simples
   sin hijos pero con restricciones y sin atributos
3. elementos complejos
   elementos con hijo y/o atributos



Estructuras

a. vacia - define un nombre del elemento y un tipo

   ```
   <xs:element name="nombre_del_elemento" type="tipo" />
   ```

b. se define el nombre y despues el tipo → no se pone el tipo del elemento como atributo.

   ```
   <xs:element name="nombre_del_elemento">
     ...
   ```

</xs:element>

Puede contener atributos cuyos valores van entre comillas:

- name. nombre del eelemnto
- type. tipo simple predefinido. ya sean los estandares o unos propios
- maxOccurs. numero máximo de veces que puede aparecer. (“0” o “unbounded”)
- minOccurs. numero minimo de veces de que puede aparece.
- ref. Para importar de otros esquemas o hacer referencia a un elemento ya declarado anteriormente en este mismo esquema.

**Sin poner maxOccurs ni minOccurs, este elemento aparece siempre exactamente una sola vez.

Tipos de datos

Restricción	Descripción
enumeration	Define los valores permitidos en una lista
fractionDigits	Especifica el número máximo de decimales permitidos. Debe ser mayor o igual a 0.
length	Especifica el número exacto de caracteres o elementos permitidos. Debe ser mayor o igual a 0.
maxExclusive	Especifica el límite superior para valores numéricos (el valor debe ser menor que este límite)
maxInclusive	Especifica el límite superior para valores numéricos (el valor debe ser menor o igual que este límite)
maxLength	Especifica o número máximo de caracteres ou valores permitidos. Debe ser maior ou igual que cero.
minExclusive	Especifica o límite inferior para valores numéricos (o valor debe ser maior que este límite)
minInclusive	Especifica o límite inferior para valores numéricos (o valor debe ser menor ou igual a este límite)
minLength	Especifica o número mínimo de caracteres ou valores permitidos. Debe ser maior ou igual que cero.
pattern	Define a secuencia exacta de caracteres permitidos.
totalDigits	Define o número exacto de díxitos permitidos. Debe ser maior que cero.
whiteSpace	Especifica como se xestionan os espazos en branco (tabs, saltos de liña, etc.)

Tipo complejo: complexType

para elementos que tienen subelementos (ele hijos) o/ y atributos

```
<xs:complexType name="nombre_del_tipo_complejo">
  <xs:sequence> <!-- sequence/all/choice -->
    ... subelementos ...
  </xs:sequence>
  ... atributos ...
</xs:complexType>
```

Orden de los elementos de complexType:

a. sequence: debene aparecer todos los elementos y en ese orden
b. all: deben aparecer todos los elementos sin importar el orden
c. choice: solo debe aparecer uno de esos elementos

attribute: para definir atributos

a. name. nombre del tipo complejo
b. mixed. puede tener valores “true” o “false”
c. type. tipo de datos con el que se especifica

Ejemplos - 2 formas

a. contiene el tipo dentro de la estructura

```
<xs:element name="contacto">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="destinatario" type="xs:string" />
      <xs:element name="remitente" type="xs:string" />
      <xs:element name="titulo" type="xs:string" />
      <xs:element name="contenido" type="xs:string" />
    </xs:sequence>
    <xs:attribute name="fecha" type="xs:date"/>
  </xs:complexType>
</xs:element>
```

b. primero define el elemento con un tipo y despues define fuera de ese tipo

```
<xs:element name="contacto" type="tipo_contacto"/>

<xs:complexType name="tipo_contacto">
  <xs:sequence>
    <xs:element name="destinatario" type="xs:string" />
    <xs:element name="remitente" type="xs:string" />
    <xs:element name="titulo" type="xs:string" />
    <xs:element name="contenido" type="xs:string" />
  </xs:sequence>
  <xs:attribute name="fecha" type="xs:date"/>
</xs:complexType>
```

***MEJOR PRACTICA. porque permite reutilizar ese tipo para otros elementos. Además los parsers toleran mejor esta estructura.

Definicion de atributos

<xs:attribute name="nombre_atributo" type="tipo_atributo" use="modificador" />

name: nombre atributo
type: tipo de atributo
use: para definir si un atributo es obligatorio o opcional.

- required. obligatorio
- default. opcional → creo??

Tipo simple: simpleType

Un tipo simple sirve para definir una serie de restricciones a un elemento o a un atributo

```
<xs:simpleType name="nombre_del_tipo_simple">
  <xs:restriction>
    ... restricciones ...
  </xs:restriction>
</xs:simpleType>
```

restriction. para rangos/patrones/enumrar posibles valores
list. para definir un tipo de lista
union. para unir tipos anteriormente definidos en uno

Puede contener atributos:
name. para poner el nombre al tipo simple

Extender un tipo

Utilizando xs:extension podemos ampliar un simpleType o complexType, añadiendo elementos o atributos extra a un tipo base definido anteriormente.

```xml
<xs:complexType name="tipo_persona">
  <xs:sequence>
    <xs:element name="nombre" type="xs:string"/>
    <xs:element name="edad" type="xs:integer"/>
  </xs:sequence>
  <xs:attribute name="id" type="xs:integer"/>
</xs:complexType>

<xs:complexType name="tipo_contacto">
  <xs:complexContent>
    <xs:extension base="tipo_persona">
      <xs:sequence>
        <xs:element name="email" type="xs:string"/>
        <xs:element name="telefono" type="xs:string"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

Estas etiquetas son parte de un lenguaje de definición de esquemas XML llamado XML Schema. En este caso, la etiqueta xs:complexType se utiliza para definir un nuevo tipo de datos complejo llamado "tipo_contacto". Este tipo de datos está compuesto por dos elementos: "email" y "telefono", que son de tipo cadena de caracteres (string).

La etiqueta xs:complexContent se utiliza para especificar la estructura compleja del tipo de datos. En este caso, se utiliza la etiqueta xs:extension para indicar que el tipo de datos "tipo_contacto" extiende otro tipo de datos llamado "tipo_persona". Esto significa que "tipo_contacto" hereda todos los elementos y atributos de "tipo_persona" y agrega los elementos "email" y "telefono" a la estructura.

La etiqueta xs:sequence se utiliza para indicar el orden en que los elementos deben aparecer en el documento XML que utiliza este esquema. En este caso, primero debe aparecer el elemento "email" y luego el elemento "telefono".

En resumen, estas etiquetas se utilizan para definir un tipo de datos complejo llamado "tipo_contacto" que representa la información de contacto de una persona y que incluye elementos para el correo electrónico y el teléfono.

Elementos sin subelmentos

Mediante simpleContent podemos definir un elemento que solo pueda contener texto y atributos, no subelementos.

Elementos vacios

Para definir un elemento vacío, que no pueda tener ni texto ni subelementos, basta con no poner ningún subelemento en la declaración del tipo:

```
<xs:complexType name="tipo_evento">
  <xs:attribute name="nombre" type="xs:string" use="required"/>
  <xs:attribute name="activo" type="xs:boolean" default="false"/>
</xs:complexType>
```

Resolviendo dudas

bol 24
error pq la etiqueta complexType del elemento descripcion debe estar fuera de la etiqueta: element y debe estar referenciado por medio de un atributo type

```
<xs:element name="listaproductos">
 <xs:complexType>

  <xs:sequence>
  <xs:element name="articulo" minOccurs="0" maxOccurs="unbounded">
   <xs:complexType>
    <xs:sequence>
     <xs:element name="codigo" type="xs:string"/>
     <xs:element name="descripcion" type="xs:string"/>
      <xs:complexType>
       <xs:attribute name="autor " type="xs:string"/>
      </xs:complexType>
    </xs:sequence>
   </xs:complexType>
  </xs:element>
  </xs:sequence>

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