Shcemas-y-DTD.pdf

frxngxrcxx

Lenguajes de marcas y sistemas de gestión de información

1º Desarrollo de Aplicaciones Multiplataforma

Estudios España

Accede al documento original

2. VALIDACIÓN MEDIANTE DTD

1. INTRODUCCIÓN

Los DTD (Document Type Definition) son un conjunto de reglas que definen la estructura y los elementos permitidos en un documento XML. Funcionan como una especificación que describe cómo debe estar organizado el contenido de un archivo XML. Un DTD puede definir:

Atributos: Las propiedades que los elementos pueden tener.

Entidades: Referencias a cadenas de texto que pueden ser reutilizadas en el documento.

2. ELEMENTOS

Qué elementos están permitidos y cómo se pueden anidar.

Elementos Simples

Los elementos simples son aquellos que no contienen otros elementos, solo texto. Se definen de la siguiente manera:

```
<!ELEMENT nombreElemento (#PCDATA)>
```

Ejemplo:

```
<!ELEMENT titulo (#PCDATA)>
```

Elementos Compuestos

Los elementos compuestos pueden contener otros elementos, además de texto. Se definen especificando qué elementos pueden estar contenidos.

```
<!ELEMENT nombreElemento (elemento1, elemento2)>
```

Ejemplo:

```
<!ELEMENT libro (titulo, autor, año)>
```

Elementos Vacíos

Los elementos vacíos no contienen contenido ni otros elementos. Se definen de la siguiente manera:

```
<!ELEMENT nombreElemento EMPTY>
```

Ejemplo:

```
<!ELEMENT portada EMPTY>
```

Elemento ANY

En un DTD, el elemento ANY se utiliza para indicar que un elemento puede contener cualquier combinación de otros elementos, texto, o incluso estar vacío.

Ejemplo:

```
  <!ELEMENT elemento ANY>
```

Aunque flexible, usar ANY puede hacer que la validación sea menos estricta, ya que no se imponen restricciones específicas sobre el contenido.

3. CARDINALIDAD DE LOS ELEMENTOS

La cardinalidad define cuántas veces un elemento puede aparecer dentro de un documento XML.

Tipos de Cardinalidad:

Sin especificador: El elemento debe aparecer exactamente una vez.

Ejemplo:

```
<!ELEMENT elemento (subelemento)>
```

?: El elemento es opcional y puede aparecer cero o una vez.

Ejemplo:

```
<!ELEMENT elemento (subelemento?)>
```

*: El elemento puede aparecer cero o más veces.

Ejemplo:

```
<!ELEMENT elemento (subelemento*)>
```

+: El elemento debe aparecer una o más veces.

Ejemplo:

```
<!ELEMENT elemento (subelemento+)>
```

Ejemplo

En este ejemplo, un libro debe tener un título, al menos un autor, y puede tener cero o más capítulos. Cada capítulo debe tener un título y puede tener cero o más párrafos.

```
<!ELEMENT libro (título, autor+, capítulo*)>
<!ELEMENT título (#PCDATA)>
<!ELEMENT autor (#PCDATA)>
<!ELEMENT capítulo (título, párrafo*)>
<!ELEMENT párrafo (#PCDATA)>
```

4. ATRIBUTOS

Los atributos se definen dentro de la declaración del elemento. Se especifica el nombre del atributo, su tipo y si es requerido o no.

```
<!ATTLIST nombreElemento nombreAtributo tipoAtributo
#REQUIRED|#IMPLIED>
```

Ejemplo:

```
<!ELEMENT libro (titulo, autor, año)>
<!ATTLIST libro
    ISBN CDATA #REQUIRED>
```

5. ENTIDADES

Las entidades son una forma de definir un conjunto de caracteres que se pueden usar en el documento. Se definen de la siguiente manera:

```
<!ENTITY nombreEntidad "valor">
```

Ejemplo:

```
<!ENTITY copyright "© 2024 Mi Empresa">
```

Los DTD pueden ser internos (definidos dentro del propio archivo XML) o externos (almacenados en un archivo separado). Aunque son útiles para la validación de la estructura de documentos XML, tienen limitaciones en cuanto a la definición de tipos de datos.

6. EJEMPLO COMPLETO DE DTD

Aquí tienes un ejemplo completo que combina todos los conceptos anteriores:

```
<!-- archivo.xml -->
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE biblioteca [
    <!ELEMENT biblioteca (libro+)>
    <!ELEMENT libro (titulo, autor, año)>
    <!ELEMENT titulo (#PCDATA)>
    <!ELEMENT autor (#PCDATA)>
    <!ELEMENT año (#PCDATA)>
    <!ATTLIST libro
        ISBN CDATA #REQUIRED>
    <!ELEMENT portada EMPTY>
    <!ENTITY copyright "© 2024 Mi Empresa">
]>
<biblioteca>
    <libro ISBN="978-3-16-148410-0">
        <titulo>El Quijote</titulo>
        <autor>Miguel de Cervantes</autor>
        <año>1605</año>
        <portada/>
    </libro>
    <libro ISBN="978-3-16-148410-1">
        <titulo>Cien años de soledad</titulo>
        <autor>Gabriel García Márquez</autor>
        <año>1967</año>
        <portada/>
    </libro>
</biblioteca>
```

8. CONCLUSIÓN

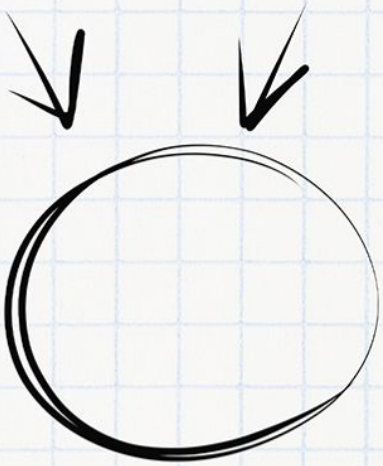
Los DTD son una herramienta útil para definir la estructura y las reglas de un documento XML. Permiten asegurar que el contenido del XML cumpla con un formato específico, lo que facilita su validación y procesamiento.

Imagínate aprobando el examen

Necesitas tiempo y concentración

Planes	PLAN TURBO	PLAN PRO	PLAN PRO+
Descargas sin publi al mes	10	40	80
Elimina el video entre descargas	✓	✓	✓
Descarga carpetas	✗	✓	✓
Descarga archivos grandes	✗	✓	✓
Visualiza apuntes online sin publi	✗	✓	✓
Elimina toda la publi web	✗	✗	✓
Precios Anual	0,99 € / mes	3,99 € / mes	7,99 € / mes

Ahora que puedes conseguirlo, ¿Qué nota vas a sacar?



WUOLAH

3. XML SCHEMAS

Los esquemas XML, también conocidos como XML Schema Definition (XSD), son una alternativa más poderosa y flexible a los DTD. Permiten definir la estructura de un documento XML de manera más detallada y precisa. Algunas de las características clave de los esquemas XML son:

Tipos de datos: Permiten definir tipos de datos complejos y simples (como cadenas, enteros, fechas, etc.).

Validación más rica: Proporcionan una validación más robusta, incluyendo restricciones sobre los valores de los atributos y elementos.

Espacios de nombres: Soportan espacios de nombres, lo que permite evitar conflictos entre elementos de diferentes fuentes.

Los esquemas XML son ampliamente utilizados en aplicaciones que requieren un alto grado de precisión y validación en la manipulación de datos XML.

1. ¿QUÉ SON LOS XML SCHEMAS?

Los XML Schemas son una forma de definir la estructura, contenido y semántica de documentos XML. A diferencia de los DTD (Document Type Definitions), los XML Schemas son más potentes y permiten especificar tipos de datos, asegurando que los datos contenidos en un documento XML sigan un formato específico.

2. UTILIDAD DE LOS XML SCHEMAS

Validación: Garantizan que un documento XML cumple con una estructura y normas específicas.

Interoperabilidad: Facilitan el intercambio de datos entre sistemas diferentes, asegurando que todos manejan el mismo formato.

Claridad y Precisión: Definen de manera precisa los tipos de datos y las relaciones entre los elementos.

Extensibilidad: Permiten la creación de formatos de datos complejos y jerárquicos.

3. DEFINICIÓN DE ELEMENTOS

En un XML Schema, los elementos se definen utilizando el elemento `<xs:element>`. Aquí se especifica el nombre del elemento y su tipo de datos.

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

4. TIPOS DE DATOS EN XML SCHEMA

Tipos Simples

- xs:string: Cadena de texto.
- xs:integer: Número entero.
- xs:decimal: Número decimal.
- xs:boolean: Valor verdadero o falso.
- xs:date: Fecha en formato YYYY-MM-DD.
- xs:time: Hora en formato HH:MM:SS.
- xs:float: Número de punto flotante de precisión simple.
- xs:double: Número de punto flotante de doble precisión.

Tipos Derivados

- xs:positiveInteger: Entero positivo.
- xs:negativeInteger: Entero negativo.
- xs:nonNegativeInteger: Entero no negativo (incluye cero).
- xs:nonPositiveInteger: Entero no positivo (incluye cero).

Tipos Compuestos

Los tipos compuestos permiten agrupar elementos y atributos.

```
<xs:complexType name="Direccion">
    <xs:sequence>
        <xs:element name="calle" type="xs:string"/>
        <xs:element name="ciudad" type="xs:string"/>
        <xs:element name="codigoPostal" type="xs:string"/>
    </xs:sequence>
</xs:complexType>
```

Definición de un Elemento Vacío

En XML Schema, un elemento de tipo vacío (o `empty`) es un elemento que no contiene ningún contenido ni elementos hijos. Se define utilizando un tipo complejo con una secuencia vacía.

XML Schema

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
    <xs:element name="elementoVacio">
        <xs:complexType>
            <xs:sequence>
                <!-- Sin elementos hijos -->
            </xs:sequence>
        </xs:complexType>
    </xs:element>
</xs:schema>
```

Documento XML

```
<elementoVacio/>
```

En este ejemplo, `elementoVacio` está definido como un elemento que no puede contener contenido ni hijos, lo que lo convierte en un elemento vacío.

ATRIBUTOS

Los atributos de un elemento se definen dentro de `<xs:attribute>` y se utilizan para especificar propiedades adicionales.

```
<xs:attribute name="id" type="xs:integer" use="required"/>
```

Tipos de Restricciones

Restricciones de Valor

1. **minInclusive / maxInclusive:** Define el valor mínimo/máximo incluido.

```
    <xs:restriction base="xs:integer">
        <xs:minInclusive value="0"/>
        <xs:maxInclusive value="100"/>
    </xs:restriction>
```

2. **minExclusive / maxExclusive:** Define el valor mínimo/máximo excluido.

3. **minLength / maxLength:** Longitud mínima/máxima para cadenas de texto.

```
    <xs:restriction base="xs:decimal">
        <xs:minExclusive value="0.0"/>
        <xs:maxExclusive value="1.0"/>
    </xs:restriction>
```

4. **pattern:** Expresión regular que la cadena debe cumplir.

```
    <xs:restriction base="xs:string">
        <xs:pattern value="\d{3}-\d{2}-\d{4}"/>
    </xs:restriction>
```

5. **enumeration:** Lista de valores permitidos.

```
    <xs:restriction base="xs:string">
        <xs:enumeration value="Rojo"/>
        <xs:enumeration value="Verde"/>
        <xs:enumeration value="Azul"/>
    </xs:restriction>
```

Restricciones de Estructura

1. **sequence:** Define que los elementos deben aparecer en un orden específico.
2. **choice:** Permite que uno y solo uno de los elementos dentro sea presente.

```
    <xs:choice>
        <xs:element name="telefono" type="xs:string"/>
        <xs:element name="email" type="xs:string"/>
    </xs:choice>
```

3. **all:** Todos los elementos deben aparecer, pero en cualquier orden.
4. **group:** Define un grupo de elementos que pueden ser reutilizados.

Documento XML

```
<persona>
    <nombre>Juan Pérez</nombre>
    <edad>30</edad>
</persona>
```

XML Schema sin Restricciones

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
    <xs:element name="persona">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="nombre" type="xs:string"/>
                <xs:element name="edad" type="xs:integer"/>
```

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

Ejemplo con Restricciones

Documento XML

```
<persona>
    <nombre>Juan Pérez</nombre>
    <edad>30</edad>
</persona>
```

XML Schema con Restricciones

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
    <xs:element name="persona">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="nombre" type="xs:string">
                    <xs:simpleType>
                        <xs:restriction base="xs:string">
                            <xs:minLength value="1"/>
                            <xs:maxLength value="50"/>
                        </xs:restriction>
                    </xs:simpleType>
                </xs:element>
                <xs:element name="edad">
                    <xs:simpleType>
                        <xs:restriction base="xs:integer">
                            <xs:minInclusive value="0"/>
                            <xs:maxInclusive value="120"/>
                        </xs:restriction>
                    </xs:simpleType>
                </xs:element>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
</xs:schema>
```

Estos ejemplos muestran cómo se define un XML Schema básico para validar un documento XML, primero sin restricciones y luego con restricciones para asegurar la validez de los datos.

Aquí tienes un ejemplo de un archivo XML que almacena información sobre vehículos, indicando si son motocicletas o coches, junto con los detalles que mencionaste:

Archivo XML (vehiculos.xml)

```
<?xml version="1.0" encoding="UTF-8"?>
<vehiculos>
    <vehiculo tipo="coche">
```

```
        <matricula>ABC1234</matricula>
        <numeroBastidor>XYZ987654321</numeroBastidor>
        <dueños>
            <dueño>Juan Pérez</dueño>
            <dueño>María López</dueño>
        </dueños>
        <cilindrada>1600</cilindrada>
    </vehiculo>
    <vehiculo tipo="motocicleta">
        <matricula>DEF5678</matricula>
        <numeroBastidor>LMN123456789</numeroBastidor>
        <dueños>
            <dueño>Carlos Ruiz</dueño>
        </dueños>
        <cilindrada>600</cilindrada>
    </vehiculo>
</vehiculos>
```

Schema XML (vehiculos.xsd)

A continuación, se proporciona un esquema XML (XSD) que valida el archivo anterior, incluyendo restricciones para los tipos de vehículos, formatos de matrícula y cilindrada.

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

    <xs:element name="vehiculos">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="vehiculo" maxOccurs="unbounded">
                    <xs:complexType>
                        <xs:sequence>
                            <xs:element name="matricula"
type="xs:string"/>
                            <xs:element name="numeroBastidor"
type="xs:string"/>
                            <xs:element name="dueños">
                                <xs:complexType>
                                    <xs:sequence>
                                        <xs:element name="dueño"
type="xs:string" maxOccurs="unbounded"/>
                                    </xs:sequence>
                                </xs:complexType>
                            </xs:element>
                            <xs:element name="cilindrada"
type="xs:int"/>
                        </xs:sequence>
                        <xs:attribute name="tipo"  use="required">
                                <xs:restriction base="xs:string">
                                    <xs:enumeration
value="motocicleta"/>
                                    <xs:enumeration value="coche"/>
                            </xs:restriction>
                        </xs:attribute>
                    </xs:complexType>
```

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

</xs:schema>
```

Ejemplo de XML y Schema XML que lo valide

```
<?xml version="1.0" encoding="UTF-8"?>
<library xmlns="http://www.example.com/library"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://www.example.com/library
library.xsd">
    <book>
        <title>El Quijote</title>
        <author>
            <firstName>Miguel</firstName>
            <lastName>Cervantes</lastName>
        </author>
        <publicationYear>1605</publicationYear>
        <genre>Ficción</genre>
        <ISBN>978-3-16-148410-0</ISBN>
        <reviews>
            <review>
                <reviewer>Juan Pérez</reviewer>
                <rating>5</rating>
                <comment>Un clásico de la literatura.</comment>
            </review>
            <review>
                <reviewer>María López</reviewer>
                <rating>4</rating>
                <comment>Muy entretenido, pero un poco
extenso.</comment>
            </review>
        </reviews>
    </book>
    <book>
        <title>Cien años de soledad</title>
        <author>
            <firstName>Gabriel</firstName>
            <lastName>García Márquez</lastName>
        </author>
        <publicationYear>1967</publicationYear>
        <genre>Realismo mágico</genre>
        <ISBN>978-3-16-148410-1</ISBN>
        <reviews>
            <review>
                <reviewer>Ana Torres</reviewer>
                <rating>5</rating>
                <comment>Una obra maestra.</comment>
            </review>
        </reviews>
```

```
    </book>
</library>
```

Ejemplo de Schema XML que valida al anterior documento XML

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

    <xs:element name="library">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="book" maxOccurs="unbounded">
                    <xs:complexType>
                        <xs:sequence>
                            <xs:element name="title"
type="xs:string"/>
                            <xs:element name="author">
                                <xs:complexType>
                                    <xs:sequence>
                                        <xs:element name="firstName"
type="xs:string"/>
                                        <xs:element name="lastName"
type="xs:string"/>
                                    </xs:sequence>
                                </xs:complexType>
                            </xs:element>
                            <xs:element name="publicationYear"
type="xs:int"/>
                            <xs:element name="genre"
type="xs:string"/>
                            <xs:element name="ISBN" type="xs:string"/>
                            <xs:element name="reviews">
                                <xs:complexType>
                                    <xs:sequence>
                                        <xs:element name="review"
maxOccurs="unbounded">
                                            <xs:complexType>
                                                <xs:sequence>
                                                    <xs:element
name="reviewer" type="xs:string"/>
                                                    <xs:element
name="rating" type="xs:int"/>
                                                    <xs:element
name="comment" type="xs:string"/>
                                                </xs:sequence>
                                            </xs:complexType>
                                        </xs:element>
                                    </xs:sequence>
                                </xs:complexType>
                            </xs:element>
                        </xs:sequence>
```

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

</xs:schema>
```

Ejemplo

Tienes el siguiente XML de ejemplo que representa información de productos:

```
<productos>
    <producto>
        <id>123</id>
        <nombre>Camisa</nombre>
        <precio>29.99</precio>
        <categoria>Ropa</categoria>
    </producto>
    <producto>
        <id>124</id>
        <nombre>Pantalón</nombre>
        <precio>49.99</precio>
        <categoria>Ropa</categoria>
    </producto>
</productos>
```

Crear un XML Schema (XSD) que restrinja el id para que sea un número de 3 dígitos exactamente, el precio para que sea un número decimal con un máximo de 100.00, y el nombre para que tenga una longitud mínima de 3 caracteres y máxima de 50.

Solución

Aquí está el archivo XSD que define las restricciones mencionadas:

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

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

    <xs:element name="productos">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="producto" maxOccurs="unbounded">
                    <xs:complexType>
                        <xs:sequence>
                            <xs:element name="id" type="idType"/>
                            <xs:element name="nombre"
type="nombreType"/>
                            <xs:element name="precio"
type="precioType"/>
                            <xs:element name="categoria"
type="xs:string"/>
                        </xs:sequence>
                    </xs:complexType>
                </xs:element>
            </xs:sequence>
        </xs:complexType>
    </xs:element>

    <xs:simpleType name="idType">
        <xs:restriction base="xs:string">
            <xs:pattern value="\d{3}"/>
        </xs:restriction>
    </xs:simpleType>

    <xs:simpleType name="nombreType">
        <xs:restriction base="xs:string">
            <xs:minLength value="3"/>
            <xs:maxLength value="50"/>
        </xs:restriction>
    </xs:simpleType>

    <xs:simpleType name="precioType">
        <xs:restriction base="xs:decimal">
            <xs:minInclusive value="0.00"/>
            <xs:maxInclusive value="100.00"/>
        </xs:restriction>
    </xs:simpleType>

</xs:schema>
```

Ejemplo

Tienes el siguiente XML que representa un catálogo de libros:

```
<catalogo>
```

```
    <libro>
        <isbn>978-3-16-148410-0</isbn>
        <titulo>Programación Avanzada</titulo>
        <autor>
            <nombre>Juan Pérez</nombre>
            <nacionalidad>MX</nacionalidad>
        </autor>
        <precio>59.95</precio>
        <publicado>2023-05-15</publicado>
    </libro>
    <libro>
        <isbn>978-1-234-56789-7</isbn>
        <titulo>Introducción a XML</titulo>
        <autor>
            <nombre>Maria López</nombre>
            <nacionalidad>ES</nacionalidad>
        </autor>
        <precio>45.50</precio>
        <publicado>2022-11-01</publicado>
    </libro>
</catalogo>
```

Debes crear un XML Schema (XSD) que cumpla con las siguientes restricciones:

El isbn debe seguir el formato estándar: "978-3-16-148410-0".

El titulo debe tener una longitud mínima de 5 caracteres y máxima de 100.

El precio debe ser un número decimal con un máximo de 200.00.

La nacionalidad debe ser un código de país de 2 letras.

La fecha de publicación debe seguir el formato de fecha ISO 8601.

Solución

Aquí está el archivo XSD que define estas restricciones:

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

    <xs:element name="catalogo">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="libro" maxOccurs="unbounded">
                    <xs:complexType>
                        <xs:sequence>
```

```
                            <xs:element name="isbn" type="isbnType"/>
                            <xs:element name="titulo"
type="tituloType"/>
                            <xs:element name="autor"
type="autorType"/>
                            <xs:element name="precio"
type="precioType"/>
                            <xs:element name="publicado"
type="xs:date"/>
                        </xs:sequence>
                    </xs:complexType>
                </xs:element>
            </xs:sequence>
        </xs:complexType>
    </xs:element>

    <xs:simpleType name="isbnType">
        <xs:restriction base="xs:string">
            <xs:pattern value="978-[0-9]{1}-[0-9]{2}-[0-9]{6}-[0-9]"/>
        </xs:restriction>
    </xs:simpleType>

    <xs:simpleType name="tituloType">
        <xs:restriction base="xs:string">
            <xs:minLength value="5"/>
            <xs:maxLength value="100"/>
        </xs:restriction>
    </xs:simpleType>

    <xs:complexType name="autorType">
        <xs:sequence>
            <xs:element name="nombre" type="xs:string"/>
            <xs:element name="nacionalidad" type="nacionalidadType"/>
        </xs:sequence>
    </xs:complexType>

    <xs:simpleType name="nacionalidadType">
        <xs:restriction base="xs:string">
            <xs:pattern value="[A-Z]{2}"/>
        </xs:restriction>
    </xs:simpleType>

    <xs:simpleType name="precioType">
        <xs:restriction base="xs:decimal">
            <xs:minInclusive value="0.00"/>
            <xs:maxInclusive value="200.00"/>
        </xs:restriction>
    </xs:simpleType>

</xs:schema>
```

Ejemplo

Tienes el siguiente XML que representa un sistema de reservas de eventos:

```
<reservas>
    <reserva>
        <codigo>EVT-2024-001</codigo>
        <evento>Concierto de Rock</evento>
        <fecha>2024-12-05T20:00:00</fecha>
        <cliente>
            <nombre>Laura Gómez</nombre>
            <email>laura.gomez@example.com</email>
        </cliente>
        <asientos reservados="2">5, 6</asientos>
    </reserva>
    <reserva>
        <codigo>EVT-2024-002</codigo>
        <evento>Obra de Teatro</evento>
        <fecha>2024-12-10T19:00:00</fecha>
        <cliente>
            <nombre>Pedro Martínez</nombre>
            <email>pedro.martinez@example.com</email>
        </cliente>
        <asientos reservados="3">10, 11, 12</asientos>
    </reserva>
</reservas>
```

Crear un XML Schema (XSD) que cumpla con las siguientes restricciones:

El codigo debe seguir el formato "EVT-YYYY-NNN".

El evento debe tener una longitud mínima de 10 caracteres y máxima de 100.

La fecha debe seguir el formato ISO 8601 para fecha y hora.

El email debe ser un formato de correo electrónico válido.

El atributo reservados de asientos debe coincidir con el número de asientos listados.

Solución

Aquí está el archivo XSD que define estas restricciones:

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

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
           xmlns:vc="http://www.w3.org/2007/XMLSchema-versioning"
           vc:minVersion="1.1">

    <xs:element name="reservas">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="reserva" maxOccurs="unbounded">
                    <xs:complexType>
                        <xs:sequence>
                            <xs:element name="codigo"
type="codigoType"/>
                            <xs:element name="evento"
type="eventoType"/>
                            <xs:element name="fecha"
type="xs:dateTime"/>
                            <xs:element name="cliente"
type="clienteType"/>
                            <xs:element name="asientos"
type="asientosType"/>
                        </xs:sequence>
                    </xs:complexType>
                </xs:element>
            </xs:sequence>
        </xs:complexType>
    </xs:element>

    <xs:simpleType name="codigoType">
        <xs:restriction base="xs:string">
            <xs:pattern value="EVT-\d{4}-\d{3}"/>
        </xs:restriction>
    </xs:simpleType>

    <xs:simpleType name="eventoType">
        <xs:restriction base="xs:string">
            <xs:minLength value="10"/>
            <xs:maxLength value="100"/>
        </xs:restriction>
    </xs:simpleType>

    <xs:complexType name="clienteType">
        <xs:sequence>
            <xs:element name="nombre" type="xs:string"/>
            <xs:element name="email" type="emailType"/>
        </xs:sequence>
    </xs:complexType>

    <xs:simpleType name="emailType">
        <xs:restriction base="xs:string">
            <xs:pattern value="^[\w-\.]+@([\w-]+\.)+[\w-]{2,4}$"/>
        </xs:restriction>
    </xs:simpleType>

    <xs:complexType name="asientosType">
        <xs:simpleContent>
            <xs:extension base="asientosListType">
```

```
                <xs:attribute name="reservados" type="xs:integer"
use="required"/>
            </xs:extension>
        </xs:simpleContent>
    </xs:complexType>

    <xs:simpleType name="asientosListType">
        <xs:restriction base="xs:string">
            <xs:pattern value="(\d+,?\s?)+"/>
        </xs:restriction>
    </xs:simpleType>

</xs:schema>
```