Tecnológico Nacional de México

Instituto Tecnológico de Culiacán

Carrera: Ingeniería en Sistemas Computacionales

Integrante:

Nombre: Olán Castro Ángel Eduardo

No.control: 21170410

Materia: Temas selectos de base de datos

Profesor: Dr. Clemente García Gerardo

Tema: XML

Unidad 4, Proyecto: XML

Fecha: miércoles, 29 de mayo del 2024

Contenido

Definición del esquema DTD (Document Type Definition)

Descripción del dominio

Me puse en el punto de vista de una empresa de streaming, como Netflix o Amazon Prime Video, las cuales, probablemente, en algún momento, tengan que enviar datos de los servicios que promocionan.

El enfoque es que, estas empresas, manejan un catálogo de entretenimiento y este catalogo consta de películas y series.

Cada película consta de un título, compañías productoras, créditos y otros atributos que las identifican, como la fecha de lanzamiento, idioma y resumen.

Las series cuentan con algunos de los atributos de las películas, además de temporadas, las cuales son transmitidas por cierto tiempo, tienen una cantidad especifica de episodios y normalmente una calificación asignada. En lugar de compañías productoras, en las series hay un enfoque mayor en autores individuales.

Es fácil ver que las series y las películas tienen mucho en común, por lo que comparten un conjunto mediano de atributos y elementos.

Formación del documento DTD

En la marcha fui creando el documento DTD, a como fui recopilando los datos que identifican a las películas y a las series. Llegué a la siguiente definición:

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

<!-- RAIZ -->
<!ELEMENT catalogo_entretenimiento (peliculas?, series?)>

<!-- PELICULAS -->
<!ELEMENT peliculas (pelicula+)>

<!ELEMENT pelicula (titulo, descripcion, pais_origen, lenguaje,
fecha_lanzamiento, generos, productoras, creditos)>
<!ELEMENT duracion (#PCDATA)>
<!ATTLIST pelicula codigo ID #REQUIRED>
<!ATTLIST pelicula duracion CDATA #REQUIRED>
<!ATTLIST pelicula presupuesto CDATA #IMPLIED>
<!ATTLIST pelicula ganancia CDATA #IMPLIED>
<!ATTLIST pelicula calificacion CDATA #IMPLIED>
```

```
<!ELEMENT productoras (productora+)>
<!ELEMENT productora (nombre, pais_origen)>

<!-- SERIES -->
<!ELEMENT series (serie+)>

<!ELEMENT serie (titulo, descripcion, pais_origen, lenguaje,
fecha_lanzamiento, temporadas, autores, generos, creditos)>
<!ATTLIST serie codigo ID #REQUIRED>

<!ELEMENT temporadas (temporada+)>
<!ELEMENT temporada (nombre, descripcion, fecha_lanzamiento)>
<!ATTLIST temporada num_episodios CDATA #REQUIRED>
<!ATTLIST temporada calificacion CDATA #IMPLIED>

<!ELEMENT autores (autor+)>
<!ELEMENT autor (nombre, genero)>

<!-- ELEMENTOS COMPARTIDOS -->
<!ELEMENT fecha_lanzamiento (anio, mes, dia)>
<!ELEMENT dia (#PCDATA)>
<!ELEMENT mes (#PCDATA)>
<!ELEMENT anio (#PCDATA)>

<!ELEMENT titulo (#PCDATA)>

<!ELEMENT descripcion (#PCDATA)>

<!ELEMENT generos (genero+)>
<!ELEMENT genero (#PCDATA)>

<!ELEMENT lenguaje (#PCDATA)>

<!ELEMENT nombre (#PCDATA)>

<!ELEMENT pais_origen (#PCDATA)>

<!ELEMENT creditos (elenco, produccion)>

<!ELEMENT elenco (actor)+>
<!ELEMENT actor (nombre, personaje)>
<!ELEMENT personaje (#PCDATA)>

<!ELEMENT produccion (operario)+>
```

```
<!ELEMENT operario (nombre, departamento, puesto)>
<!ELEMENT departamento (#PCDATA)>
<!ELEMENT puesto (#PCDATA)>
```

Archivo XML

Creación del archivo XML

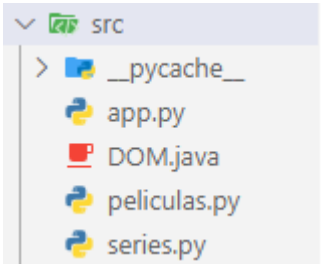
Pude obtener los datos necesarios utilizando la API de la página web The Movie Database (https://www.themoviedb.org/). Esta permite realizar distintas consultas.

Específicamente, utilicé 3 tipos de consultas:

- General. Esta permite obtener datos generales de una gran cantidad de series o películas. Normalmente, una consulta te permite obtener datos de 20, pues se tiene una organización por páginas y ese es el máximo que detecté.
- Específica. La información general no me bastaba, por lo que realizaba por cada serie o película una consulta específica, a partir de la cual obtenía los datos restantes.
- Créditos. Dentro de la información específica, los administradores de este servicio, decidieron no colocar los créditos, por lo que tuve que comunicarme con otro endpoint para realizar dicha consulta y obtener los datos.

Ahora, para agilizar este proceso, y engrandecer mis conocimientos, utilicé Python.

Esta fue la estructura que seguí.



Partiendo desde el app.py

```
import xml.etree.cElementTree as ET
from peliculas import meter_peliculas
from series import meter_series

def main():
  root = ET.Element('catalogo_entretenimiento')
```

```
  meter_peliculas(root)
  meter_series(root)

  ET.indent(root)
  with open('data/catalogo.xml', 'wb') as f:
    f.write('<?xml version="1.0" encoding="UTF-8" ?>\n<!DOCTYPE
catalogo_entretenimiento SYSTEM "catalogo.dtd">'.encode('utf8'))
    ET.ElementTree(root).write(f, 'utf-8')
main()
```

Al igual que en java, en Python se tiene una librería para facilitar la construcción de un documento XML, en este caso, se maneja por nodos, por lo que, para construir todo el documento, fui, por cada registro de la base de datos, agregando sus nodos manualmente, la mayor carga fue el mapeo.

Como se puede observar, partimos de una raíz, la cual será el catálogo, de la cual saldrán los nodos de películas y series. Un poco más abajo se verán esas clases.

Afortunadamente, este servicio web me pudo proveer toda la información que requería para satisfacer los requerimientos del DTD.

Las clases que restan son las usadas para agregar las películas y series.

Películas.py

```
import xml.etree.ElementTree as ET
import requests
import json

def meter_peliculas(root):
  root_peliculas = ET.SubElement(root, 'peliculas')

  for pagina in range(5):
    meter_peliculas_pagina(pagina + 1, root_peliculas)
  print("Peliculas acaba")

def meter_peliculas_pagina(pagina, root):
  peliculas_json = get_peliculas_json(pagina)
  peliculas_dict = json.loads(peliculas_json)

  for pelicula in peliculas_dict["results"]:
    meter_pelicula(pelicula, root)

def meter_pelicula(pelicula_general, root):
  id_pelicula = pelicula_general["id"]
```

```
  pelicula_json = get_pelicula_json(id_pelicula)
  pelicula_dict = json.loads(pelicula_json)

  _codigo = "a" + str(id_pelicula)
  _duracion = pelicula_dict["runtime"]
  _presupuesto = pelicula_dict["budget"]
  _ganancia =  pelicula_dict["revenue"]
  _calificacion = pelicula_dict["vote_average"]
  nodo_pelicula = ET.SubElement(root,
                                'pelicula',
                                codigo = str(_codigo),
                                duracion = str(_duracion),
                                presupuesto = str(_presupuesto),
                                ganancia = str(_ganancia),
                                calificacion = str(_calificacion))

  ET.SubElement(nodo_pelicula, 'titulo').text = pelicula_dict["title"]
  ET.SubElement(nodo_pelicula, 'descripcion').text =
pelicula_dict["overview"]
  ET.SubElement(nodo_pelicula, 'pais_origen').text =
pelicula_dict["origin_country"][0]
  ET.SubElement(nodo_pelicula, 'lenguaje').text =
pelicula_dict["original_language"]

  fecha_lanzamiento = str(pelicula_dict["release_date"])
  anio, mes, dia = fecha_lanzamiento.split("-")
  nodo_fecha = ET.SubElement(nodo_pelicula, 'fecha_lanzamiento')
  ET.SubElement(nodo_fecha, 'anio').text = anio
  ET.SubElement(nodo_fecha, 'mes').text = mes
  ET.SubElement(nodo_fecha, 'dia').text = dia

  nodo_generos = ET.SubElement(nodo_pelicula, 'generos')
  for genero in pelicula_dict["genres"]:
    ET.SubElement(nodo_generos, 'genero').text = genero["name"]

  nodo_productoras = ET.SubElement(nodo_pelicula, 'productoras')
  for productora in pelicula_dict["production_companies"]:
    nodo_productora = ET.SubElement(nodo_productoras, 'productora')
    ET.SubElement(nodo_productora, 'nombre').text = productora["name"]
    ET.SubElement(nodo_productora, 'pais_origen').text =
productora["origin_country"]

  creditos_json = get_creditos_json(id_pelicula)
  creditos_dict = json.loads(creditos_json)
  nodo_creditos = ET.SubElement(nodo_pelicula, 'creditos')
```

```
  nodo_elenco = ET.SubElement(nodo_creditos, 'elenco')
  for actor in creditos_dict["cast"]:
    nodo_actor = ET.SubElement(nodo_elenco, 'actor')
    ET.SubElement(nodo_actor, 'nombre').text = actor["name"]
    ET.SubElement(nodo_actor, 'personaje').text = actor["character"]

  nodo_produccion = ET.SubElement(nodo_creditos, 'produccion')
  for operario in creditos_dict["crew"]:
    nodo_operario = ET.SubElement(nodo_produccion, 'operario')
    ET.SubElement(nodo_operario, 'nombre').text = operario["name"]
    ET.SubElement(nodo_operario, 'departamento').text =
operario["department"]
    ET.SubElement(nodo_operario, 'puesto').text = operario["job"]

def get_creditos_json(id):
  url = f"https://api.themoviedb.org/3/movie/{id}/credits"

  headers = {
      "accept": "application/json",
      "Authorization": "Bearer
eyJhbGciOiJIUzI1NiJ9.eyJhdWQiOiI1ZjYwMjMxZWM0ZTY0Zjk5OTQxOWUyMWUyMDlmYjU2MiI
sInN1YiI6IjY2NTUyOTFkMWM1MjljNjU5MTExMTI5MiIsInNjb3BlcyI6WyJhcGlfcmVhZCJdLCJ
2ZXJzaW9uIjoxfQ.Qv-5dHpCCvVHMkUwEbGEL5rUlpBenfZLjhmLpzWbDuw"
  }

  response = requests.get(url, headers=headers)

  return response.text

def get_pelicula_json(id):
  url = f"https://api.themoviedb.org/3/movie/{id}"

  headers = {
      "accept": "application/json",
      "Authorization": "Bearer
eyJhbGciOiJIUzI1NiJ9.eyJhdWQiOiI1ZjYwMjMxZWM0ZTY0Zjk5OTQxOWUyMWUyMDlmYjU2MiI
sInN1YiI6IjY2NTUyOTFkMWM1MjljNjU5MTExMTI5MiIsInNjb3BlcyI6WyJhcGlfcmVhZCJdLCJ
2ZXJzaW9uIjoxfQ.Qv-5dHpCCvVHMkUwEbGEL5rUlpBenfZLjhmLpzWbDuw"
  }

  response = requests.get(url, headers=headers)

  return response.text
```

```
def get_peliculas_json(pagina) :
  url = f"https://api.themoviedb.org/3/discover/movie?page={pagina}"

  headers = {
      "accept": "application/json",
      "Authorization": "Bearer
eyJhbGciOiJIUzI1NiJ9.eyJhdWQiOiI1ZjYwMjMxZWM0ZTY0Zjk5OTQxOWUyMWUyMDlmYjU2MiI
sInN1YiI6IjY2NTUyOTFkMWM1MjljNjU5MTExMTI5MiIsInNjb3BlcyI6WyJhcGlfcmVhZCJdLCJ
2ZXJzaW9uIjoxfQ.Qv-5dHpCCvVHMkUwEbGEL5rUlpBenfZLjhmLpzWbDuw"
  }

  response = requests.get(url, headers=headers)

  return response.text
```

series.py

```
import xml.etree.ElementTree as ET
import requests
import json

def meter_series(root):
  root_series = ET.SubElement(root, 'series')

  for pagina in range(5):
    meter_series_pagina(pagina + 1, root_series)
  print("series acaba")

def meter_series_pagina(pagina, root):
  series_json = get_series_json(pagina)
  series_dict = json.loads(series_json)

  for serie in series_dict["results"]:
    meter_serie(serie, root)

def meter_serie(serie_general, root):
  id_serie = serie_general["id"]
  serie_json = get_serie_json(id_serie)
  serie_dict = json.loads(serie_json)

  _codigo = "a" + str(id_serie)
  nodo_serie = ET.SubElement(root,
                             'serie',
                             codigo = str(_codigo))
```

```
  ET.SubElement(nodo_serie, 'titulo').text = serie_dict["name"]
  ET.SubElement(nodo_serie, 'descripcion').text = serie_dict["overview"]
  ET.SubElement(nodo_serie, 'pais_origen').text =
serie_dict["origin_country"][0]
  ET.SubElement(nodo_serie, 'lenguaje').text =
serie_dict["original_language"]

  fecha_lanzamiento = str(serie_dict["first_air_date"])
  anio, mes, dia = fecha_lanzamiento.split("-")
  nodo_fecha = ET.SubElement(nodo_serie, 'fecha_lanzamiento')
  ET.SubElement(nodo_fecha, 'anio').text = anio
  ET.SubElement(nodo_fecha, 'mes').text = mes
  ET.SubElement(nodo_fecha, 'dia').text = dia

  nodo_temporadas = ET.SubElement(nodo_serie, 'temporadas')
  for temporada in serie_dict["seasons"]:
    _num_episodios = temporada["episode_count"]
    _calificacion = temporada["vote_average"]
    nodo_temporada = ET.SubElement(nodo_temporadas,
                                   'temporada',
                                    num_episodios = str(_num_episodios),
                                    calificacion = str(_calificacion))
    ET.SubElement(nodo_temporada, 'nombre').text = temporada["name"]
    ET.SubElement(nodo_temporada, 'descripcion').text =
temporada["overview"]
    nodo_temporada_fecha = ET.SubElement(nodo_temporada,
                                         'fecha_lanzamiento')
    if (temporada["air_date"] == None):
      ET.SubElement(nodo_temporada_fecha, 'anio').text = anio
      ET.SubElement(nodo_temporada_fecha, 'mes').text = mes
      ET.SubElement(nodo_temporada_fecha, 'dia').text = dia
    else:
      anio, mes, dia = fecha_lanzamiento.split("-")
      ET.SubElement(nodo_temporada_fecha, 'anio').text = anio
      ET.SubElement(nodo_temporada_fecha, 'mes').text = mes
      ET.SubElement(nodo_temporada_fecha, 'dia').text = dia

  nodo_autores = ET.SubElement(nodo_serie, 'autores')
  for autor in serie_dict["created_by"]:
    nodo_autor = ET.SubElement(nodo_autores, 'autor')
    ET.SubElement(nodo_autor, 'nombre').text = autor["name"]
    ET.SubElement(nodo_autor, 'genero').text = str(autor["gender"])

  nodo_generos = ET.SubElement(nodo_serie, 'generos')
  for genero in serie_dict["genres"]:
```

```
    ET.SubElement(nodo_generos, 'genero').text = genero["name"]

  creditos_json = get_creditos_json(id_serie)
  creditos_dict = json.loads(creditos_json)
  nodo_creditos = ET.SubElement(nodo_serie, 'creditos')

  nodo_elenco = ET.SubElement(nodo_creditos, 'elenco')
  for actor in creditos_dict["cast"]:
    nodo_actor = ET.SubElement(nodo_elenco, 'actor')
    ET.SubElement(nodo_actor, 'nombre').text = actor["name"]
    ET.SubElement(nodo_actor, 'personaje').text = actor["character"]

  nodo_produccion = ET.SubElement(nodo_creditos, 'produccion')
  for operario in creditos_dict["crew"]:
    nodo_operario = ET.SubElement(nodo_produccion, 'operario')
    ET.SubElement(nodo_operario, 'nombre').text = operario["name"]
    ET.SubElement(nodo_operario, 'departamento').text =
operario["department"]
    ET.SubElement(nodo_operario, 'puesto').text = operario["job"]

def get_creditos_json(id):
  url = f"https://api.themoviedb.org/3/tv/{id}/credits"

  headers = {
      "accept": "application/json",
      "Authorization": "Bearer
eyJhbGciOiJIUzI1NiJ9.eyJhdWQiOiI1ZjYwMjMxZWM0ZTY0Zjk5OTQxOWUyMWUyMDlmYjU2MiI
sInN1YiI6IjY2NTUyOTFkMWM1MjljNjU5MTExMTI5MiIsInNjb3BlcyI6WyJhcGlfcmVhZCJdLCJ
2ZXJzaW9uIjoxfQ.Qv-5dHpCCvVHMkUwEbGEL5rUlpBenfZLjhmLpzWbDuw"
  }

  response = requests.get(url, headers=headers)

  return response.text

def get_serie_json(id):
  url = f"https://api.themoviedb.org/3/tv/{id}"

  headers = {
      "accept": "application/json",
      "Authorization": "Bearer
eyJhbGciOiJIUzI1NiJ9.eyJhdWQiOiI1ZjYwMjMxZWM0ZTY0Zjk5OTQxOWUyMWUyMDlmYjU2MiI
sInN1YiI6IjY2NTUyOTFkMWM1MjljNjU5MTExMTI5MiIsInNjb3BlcyI6WyJhcGlfcmVhZCJdLCJ
2ZXJzaW9uIjoxfQ.Qv-5dHpCCvVHMkUwEbGEL5rUlpBenfZLjhmLpzWbDuw"
  }
```

```
  response = requests.get(url, headers=headers)

  return response.text

def get_series_json(pagina) :
  url = f"https://api.themoviedb.org/3/discover/tv?page={pagina}"

  headers = {
      "accept": "application/json",
      "Authorization": "Bearer
eyJhbGciOiJIUzI1NiJ9.eyJhdWQiOiI1ZjYwMjMxZWM0ZTY0Zjk5OTQxOWUyMWUyMDlmYjU2MiI
sInN1YiI6IjY2NTUyOTFkMWM1MjljNjU5MTExMTI5MiIsInNjb3BlcyI6WyJhcGlfcmVhZCJdLCJ
2ZXJzaW9uIjoxfQ.Qv-5dHpCCvVHMkUwEbGEL5rUlpBenfZLjhmLpzWbDuw"
  }

  response = requests.get(url, headers=headers)

  return response.text
```

Los métodos que contienen la subcadena “json” en su nombre, son los que se comunican directamente con los endpoints de la API. Lo más difícil, como establecí antes, fue el mapeo de los datos y dentro de ello, la creación del árbol.

Cabe decir que Python facilitó, una vez más, este proceso, pues tiene una librería que permite hacer un diccionario con el contenido de un JSON.

Mostrar el contenido del archivo XML

catalogo.xml

data > catalogo.xml

```xml
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE catalogo_entretenimiento SYSTEM "catalogo.dtd"><catalogo_entretenimiento>
  <peliculas>
    <pelicula codigo="a823464" duracion="115" presupuesto="150000000" ganancia="558503759" calificacion="7.275">
      <titulo>Godzilla x Kong: The New Empire</titulo>
      <descripcion>Following their explosive showdown, Godzilla and Kong must reunite against a colossal undiscovered threat hidden
      <pais_origen>US</pais_origen>
      <lenguaje>en</lenguaje>
      <fecha_lanzamiento>
        <anio>2024</anio>
        <mes>03</mes>
        <dia>27</dia>
      </fecha_lanzamiento>
      <generos>
        <genero>Science Fiction</genero>
        <genero>Action</genero>
        <genero>Adventure</genero>
      </generos>
      <productoras>
        <productora>
          <nombre>Legendary Pictures</nombre>
          <pais_origen>US</pais_origen>
        </productora>
      </productoras>
      <creditos>
        <elenco>
          <actor>
            <nombre>Rebecca Hall</nombre>
            <personaje>Dr. Ilene Andrews</personaje>
          </actor>
          <actor>
            <nombre>Brian Tyree Henry</nombre>
            <personaje>Bernie Hayes</personaje>
          </actor>
          <actor>
            <nombre>Dan Stevens</nombre>
            <personaje>Trapper</personaje>
          </actor>
          <actor>
```

data > catalogo.xml

```xml
<!DOCTYPE catalogo_entretenimiento SYSTEM "catalogo.dtd"><catalogo_entretenimiento>
  <series>
    <serie codigo="a2734">
      <temporadas>
        <temporada num_episodios="6" calificacion="0.0">
          <fecha_lanzamiento>
            <anio>1999</anio>
            <mes>09</mes>
            <dia>20</dia>
          </fecha_lanzamiento>
        </temporada>
        <temporada num_episodios="22" calificacion="7.2">
          <nombre>Season 1</nombre>
          <descripcion />
          <fecha_lanzamiento>
            <anio>1999</anio>
            <mes>09</mes>
            <dia>20</dia>
          </fecha_lanzamiento>
        </temporada>
        <temporada num_episodios="21" calificacion="7.8">
          <nombre>Season 2</nombre>
          <descripcion />
          <fecha_lanzamiento>
            <anio>1999</anio>
            <mes>09</mes>
            <dia>20</dia>
          </fecha_lanzamiento>
        </temporada>
        <temporada num_episodios="23" calificacion="7.8">
          <nombre>Season 3</nombre>
          <descripcion />
          <fecha_lanzamiento>
            <anio>1999</anio>
            <mes>09</mes>
            <dia>20</dia>
          </fecha_lanzamiento>
        </temporada>
        <temporada num_episodios="25" calificacion="7.4">
```

Validar documento XML

Bien formado (no checa las restricciones de XML)

Programa JAVA

La falla radicaba en que, al tener el manejador de errores por defecto, este "consumía" la excepción y por lo tanto, no se arrojaba y nunca se recibía en el método main, por lo que el flujo del programa seguía perfectamente.

La solución a esto: crear mi propio manejador de errores.

```
import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;
import org.xml.sax.ErrorHandler;
import org.xml.sax.SAXException;
import org.xml.sax.SAXParseException;

public class XMLBIENFORMADO {
    public static void main(String [] args) {
       if( args.length != 1) {
          System.out.println("error, debe de proporcionar el nombre del
archivo");
          return;
       }
       String arch=args[0];
       System.out.println("*****VALIDANDO SI ESTÁ BIEN FORMADOS EL ARCHIVO
"+arch);
       try {
          DocumentBuilderFactory dbFactory =
DocumentBuilderFactory.newInstance();

          DocumentBuilder dBuilder = dbFactory.newDocumentBuilder();
          dBuilder.setErrorHandler(new MiErrorHandler());
          dBuilder.parse(arch);

          System.out.println("********fin del analisis, documento bien
formado************");
    }
       catch (Exception e) {
         System.out.println("********fin del analisis, documento mal
formado************");
          System.out.println(e);
          return;
       }
```

```
    }
}

class MiErrorHandler implements ErrorHandler {
  @Override
  public void warning(SAXParseException exception) throws SAXException {
    throw new SAXException(exception.getMessage());
  }

  @Override
  public void error(SAXParseException exception) throws SAXException {
    throw new SAXException(exception.getMessage());
  }

  @Override
  public void fatalError(SAXParseException exception) throws SAXException {
    throw new SAXException(exception.getMessage());
  }
}
```

Resultado

```
PS D:\ProyectoXML> java .\src\XMLBIENFORMADO.java .\data\catalogo.xml
*****VALIDANDO SI ESTÁ BIEN FORMADOS EL ARCHIVO .\data\catalogo.xml
********fin del analisis, documento bien formado**********
```

Explicación

El archivo está bien formado, es decir, se cumplen las reglas mínimas del XML, como que se cierre toda etiqueta que se abre.

Válido

Programa y herramienta Oxígeno

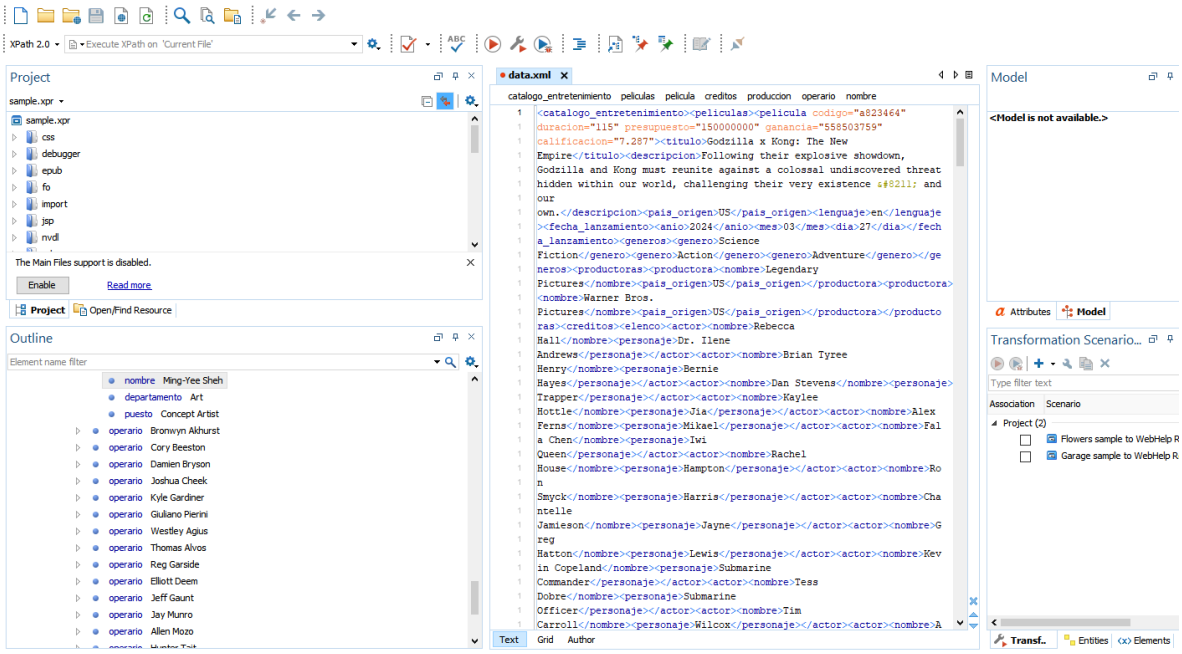
Al igual que con lo anterior, implementé el manejador de errores, el cual solucionó la problemática.

```
import org.w3c.dom.Document;
import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;
import org.xml.sax.ErrorHandler;
import org.xml.sax.SAXException;
import org.xml.sax.SAXParseException;

public class XMLVALIDO {
    public static void main(String [] args) {
```

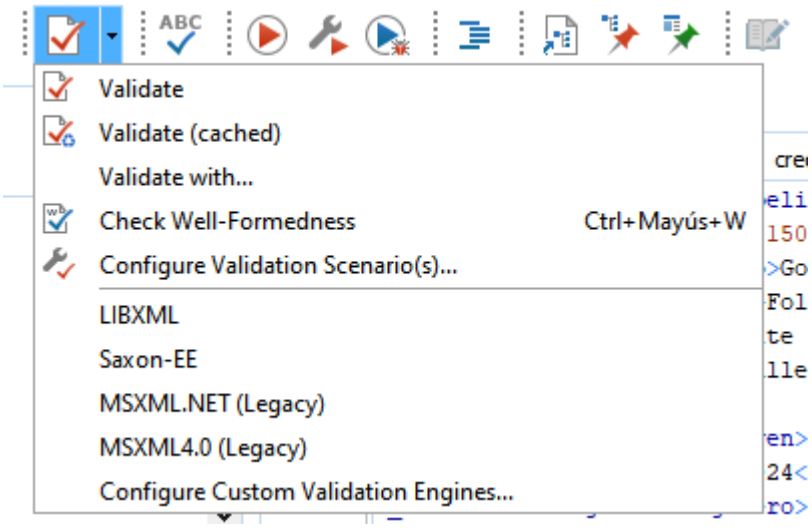
```
        if( args.length != 1) {
           System.out.println("error, debe de proporcionar el nombre del
archivo");
           return;
        }
        String arch=args[0];
        System.out.println("*****VALIDADNDO, SI ES VALIDO EL ARCHIVO "+arch);
        try {
           DocumentBuilderFactory dbFactory =
DocumentBuilderFactory.newInstance();
           dbFactory.setValidating(true);
           DocumentBuilder dBuilder = dbFactory.newDocumentBuilder();
           dBuilder.setErrorHandler(new MiErrorHandler());
           Document doc = dBuilder.parse(arch);

           System.out.println("********fin del analisis, documento
valido***********");
     }
        catch (Exception e) {
          System.out.println("********fin del analisis, documento
invalido***********");
           System.out.println(e);
           return;
        }
     }
}

class MiErrorHandler implements ErrorHandler {
   @Override
   public void warning(SAXParseException exception) throws SAXException {
     throw new SAXException(exception.getMessage());
   }

   @Override
   public void error(SAXParseException exception) throws SAXException {
     throw new SAXException(exception.getMessage());
   }

   @Override
   public void fatalError(SAXParseException exception) throws SAXException {
     throw new SAXException(exception.getMessage());
   }
 }
```

Para la herramienta Oxígeno, bastaron pocos pasos:

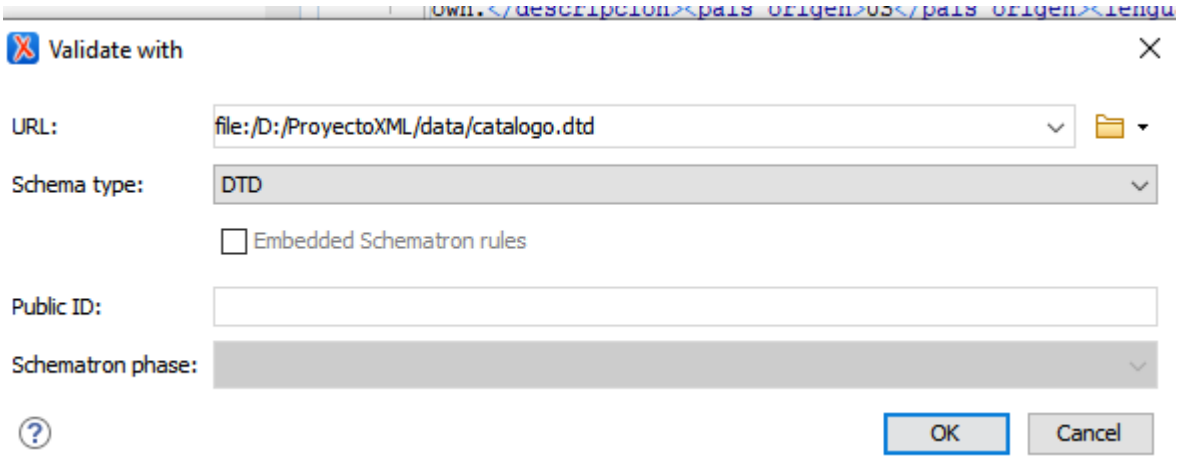
Abrir el archivo



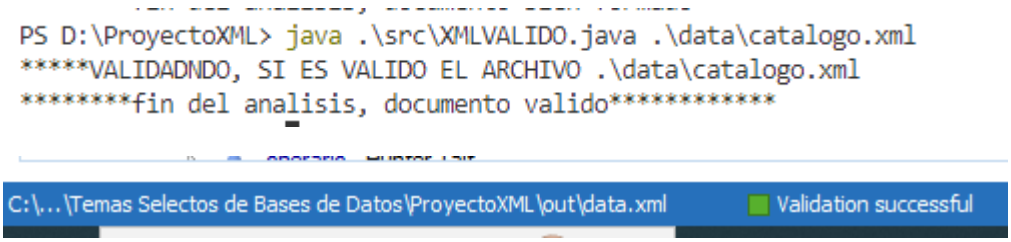
Seleccionar de que forma se validará el archivo. La opción validate with es la indicada



Seleccionar el archivo DTD



Resultado



Explicación

Tanto el programa en java como la herramienta Oxígeno determinaron que el archivo XML es válido, es decir, que dentro de él, se cumplen todas las reglas que fueron determinadas en el archivo DTD de definición.

Mostrar contenido del documento XML

Programa JAVA

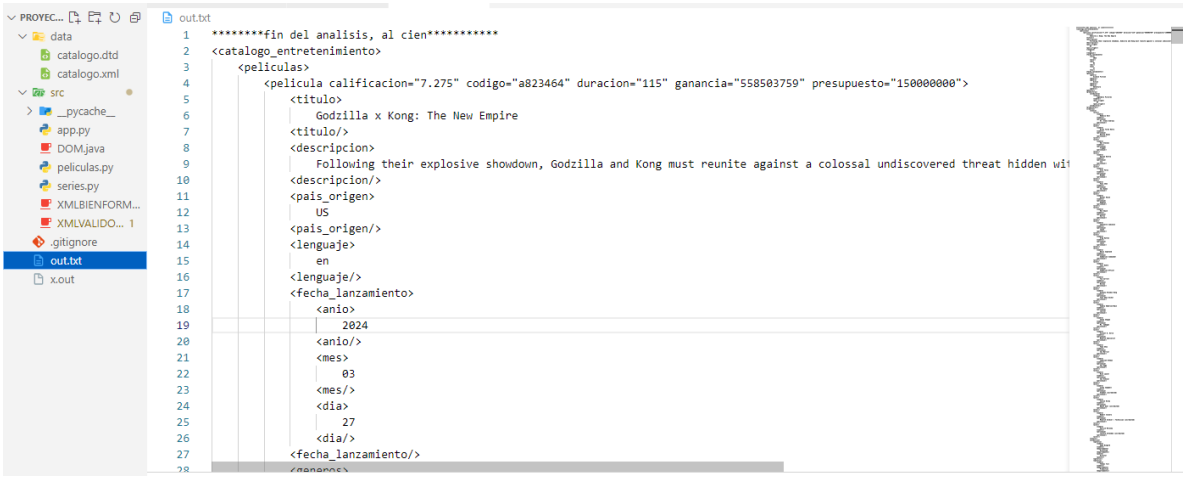
```
import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;

import org.w3c.dom.Document;
import org.w3c.dom.NamedNodeMap;
import org.w3c.dom.Node;
import org.w3c.dom.NodeList;
import org.xml.sax.ErrorHandler;
import org.xml.sax.SAXException;
import org.xml.sax.SAXParseException;

public class DOM {
  public static void main(String[] args) {
    String file = args[0];
    try {
      DocumentBuilderFactory dbFactory =
DocumentBuilderFactory.newInstance();
      dbFactory.setValidating(true);

      DocumentBuilder dBuilder = dbFactory.newDocumentBuilder();
      dBuilder.setErrorHandler(new MiErrorHandler());
      Document document = dBuilder.parse(file);

      System.out.println("********fin del analisis, al cien**********");

      processDOM(document.getDocumentElement(), 0);
    } catch (Exception e) {
      System.out.println("********fin del analisis, ya valio***********");
      System.out.println(e);
      return;
    }
  }

  public static void procesarDOM(Node node, int espacioIzquierda) {
    short tipo = node.getNodeType();

    if (tipo == Node.TEXT_NODE) {
      String contenido = node.getTextContent();
      if (!contenido.isBlank()) {
         System.out.println(" ".repeat(espacioIzquierda) + contenido);
      }
```

```
      return;
    }

    if (tipo != Node.ELEMENT_NODE) {
      return;
    }


    System.out.print(" ".repeat(espacioIzquierda) + "<" +
node.getNodeName());

    if (node.hasAttributes()) {
       NamedNodeMap mapaAtributos = node.getAttributes();
       for (int i = 0; i < mapaAtributos.getLength(); i++) {
          Node atributo = mapaAtributos.item(i);
          System.out.print(" " + atributo.getNodeName() + "=\"" +
atributo.getNodeValue() + "\"");
         }
   }

   boolean tieneHijos = node.hasChildNodes();
    System.out.println(tieneHijos ? ">" : "/>");

    NodeList hijos = node.getChildNodes();
    for (int i = 0; i < hijos.getLength(); i++) {
      procesarDOM(hijos.item(i), espacioIzquierda + 4);
    }

    if (tieneHijos) {
      System.out.println(" ".repeat(espacioIzquierda) + "<" +
node.getNodeName() + "/>");
    }
  }
class MiErrorHandler implements ErrorHandler {
   @Override
   public void warning(SAXParseException exception) throws SAXException {
     throw new SAXException(exception.getMessage());
   }

   @Override
   public void error(SAXParseException exception) throws SAXException {
     throw new SAXException(exception.getMessage());
   }

   @Override
```

```
    public void fatalError(SAXParseException exception) throws SAXException {
      throw new SAXException(exception.getMessage());
    }
  }
```

Checamos

```
PS D:\ProyectoXML> java .\src\DOM.java .\data\catalogo.xml > out.txt
```



Mostrar los datos del archivo XML

Programa JAVA

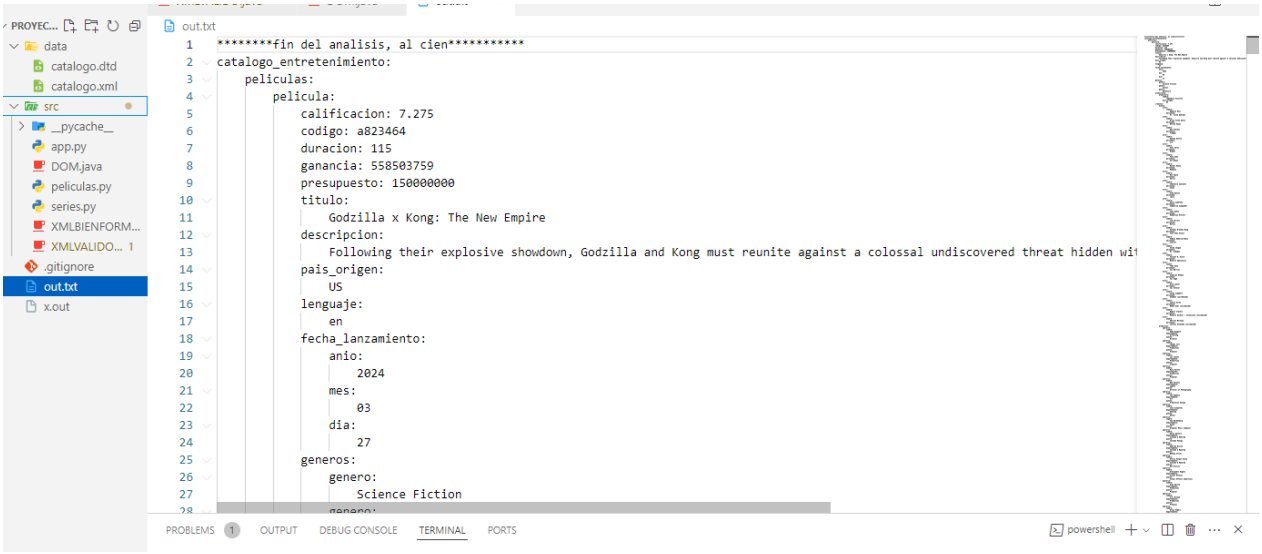
```
import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;

import org.w3c.dom.Document;
import org.w3c.dom.NamedNodeMap;
import org.w3c.dom.Node;
import org.w3c.dom.NodeList;
import org.xml.sax.ErrorHandler;
import org.xml.sax.SAXException;
import org.xml.sax.SAXParseException;

public class DOM {
  public static void main(String[] args) {
    String file = args[0];
    try {
      DocumentBuilderFactory dbFactory =
DocumentBuilderFactory.newInstance();
      dbFactory.setValidating(true);

      DocumentBuilder dBuilder = dbFactory.newDocumentBuilder();
```

```
      dBuilder.setErrorHandler(new MiErrorHandler());
      Document document = dBuilder.parse(file);

      System.out.println("********fin del analisis, al cien***********");

      muestraContenido(document.getDocumentElement(), 0);
    } catch (Exception e) {
      System.out.println("********fin del analisis, ya valio***********");
      System.out.println(e);
      return;
    }
  }

  public static void muestraContenido(Node nodo, int espacioIzquierda) {
    short tipo = nodo.getNodeType();

    if (tipo == Node.TEXT_NODE) {
      String contenido = nodo.getTextContent();
      if (!contenido.isBlank()) {
         System.out.println(" ".repeat(espacioIzquierda) + contenido);
      }
      return;
    }

    if (tipo != Node.ELEMENT_NODE) {
      return;
    }

    System.out.println(" ".repeat(espacioIzquierda) + nodo.getNodeName() +
":");

    if (nodo.hasAttributes()) {
      NamedNodeMap nodoNombrado = nodo.getAttributes();
      for (int i = 0; i < nodoNombrado.getLength(); i++) {
        Node atributo = nodoNombrado.item(i);
        System.out.println(" ".repeat(espacioIzquierda + 4) +
atributo.getNodeName() + ": " + atributo.getNodeValue());
      }
    }

    NodeList hijos = nodo.getChildNodes();
    for (int i = 0; i < hijos.getLength(); i++) {
      muestraContenido(hijos.item(i), espacioIzquierda + 4);
    }
  }
```

```
}

class MiErrorHandler implements ErrorHandler {
   @Override
   public void warning(SAXParseException exception) throws SAXException {
     throw new SAXException(exception.getMessage());
   }

   @Override
   public void error(SAXParseException exception) throws SAXException {
     throw new SAXException(exception.getMessage());
   }

   @Override
   public void fatalError(SAXParseException exception) throws SAXException {
     throw new SAXException(exception.getMessage());
   }
 }
```

Checamos

```
PS D:\ProyectoXML> java .\src\DOM.java .\data\catalogo.xml > out.txt
```



Herramienta XML Oxígeno