DESARROLLO DIRIGIDO POR MODELOS EN EL ÁMBITO MUSEÍSTICO

MODEL DRIVEN DEVELOPMENT IN THE MUSEUM ENVIRONMENT

NOTA FINAL: 8
TRABAJO FIN DE GRADO
CURSO 2023-2024

AUTOR
PAULA GONZÁLEZ HERERRA

DIRECTOR
RUBÉN FUENTES FERNÁNDEZ

GRADO EN INGENIERÍA INFORMÁTICA
FACULTAD DE INFORMÁTICA
UNIVERSIDAD COMPLUTENSE DE MADRID

DESARROLLO DIRIGIDO POR MODELOS EN EL ÁMBITO MUSEÍSTICO
MODEL DRIVEN DEVELOPMENT IN THE MUSEUM ENVIRONMENT

TRABAJO DE FIN DE GRADO EN INGENIERÍA INFORMÁTICA

AUTOR
PAULA GONZÁLEZ HERRERA

DIRECTOR
RUBÉN FUENTES FERNÁNDEZ

CONVOCATORIA: ENERO 2025

GRADO EN INGENIERÍA INFORMÁTICA
FACULTAD DE INFORMÁTICA
UNIVERSIDAD COMPLUTENSE DE MADRID

17 DE ENERO 2025

DEDICATORIA

A mi tío Pedro y a mis padres por estar apoyándome durante el proyecto.

AGRADECIMIENTOS

A Rubén Fuentes-Fernández por el tiempo, ayuda y la paciencia invertida en este trabajo, no habría sido posible sin él.

RESUMEN

Desarrollo Dirigido por Modelos en el ámbito museístico

Los museos están realizando importantes esfuerzos por hacer sus colecciones accesibles a la mayor variedad posible de públicos. La diversidad (ejemplo: sensorial, cognitiva, de formación o cultural) hace necesario adaptar la comunicación para facilitar a los públicos la comprensión y el disfrute de sus colecciones.

Dependiendo del público y de los museos se necesitan una diversidad de intervenciones que incluyen, por ejemplo, diferentes etiquetas de las piezas, guías audibles, recorridos dependientes de las capacidades motoras, generación de modelos 3D que puedan ser tocados o aplicaciones móviles de realidad aumentada. Con los presupuestos limitados, es necesario posibilitar una aproximación incremental y maximizar la cantidad y calidad de los productos obtenidos.

En este proyecto se explora el MDE (Ingeniería Dirigida por Modelos) como ayuda a la generación de diferentes artefactos a partir de los mismos datos de una forma rápida y reproducible. Para ello se han aplicado diferentes técnicas del ámbito MDE, como la transformación de un modelo XML de datos en artefactos HTML y JSON, utilizando los marcos Xtend, Xtext y EMF.

Palabras clave

Museos, Web Scraping, Xtext y Xtend, Eclipse Modeling Framework.

ABSTRACT

Model Driven Development in the museum environment

Museums are making significant efforts to make their collections accessible to the widest possible range of audiences. Diversity (e.g. sensory, cognitive, educational or cultural diversity) makes it necessary to adapt the museums to make it easier for audiences to understand and enjoy the collections.

Depending on the audience and the museums, a variety of interventions are needed, including, for example, different cartouches of the pieces, audible guides, motor-dependent tours, generation of 3D models that can be touched, or augmented reality mobile apps. With limited budgets, it is necessary to enable an incremental approach and maximize the quantity and quality of the products obtained.

The goal of this project was testing that MDE (Model Driven Engineering) significantly helps to generate different artifacts from the same data in a fast and reproducible way.

In this project, different techniques of the MDE field have been applied, such as the transformation of an XML data model into HTML and JSON artifacts, using the frameworks Xtend, Xtext and EMF tools.

Keywords: Museums, Web Scraping, Xtext and Xtend,Eclipse Modeling Framework.

ÍNDICE DE CONTENIDOS

ÍNDICE DE FIGURAS

Capítulo 1 - Introducción

A medida que las tecnologías avanzan en la actualidad, los museos deben adaptarse a dichos avances para poder generar ingresos y alcanzar un número alto de visitas, lo que supone un gran reto para muchos de ellos, sobre todo para aquellos museos que tienen recursos limitados y escasos conocimientos de programación. Además, de que en la actualidad hay una gran parte de usuarios que prefiere dichas tecnologías novedosas como la realidad aumentada o recorridos virtuales para aumentar su información cultural. Este proyecto da pie a que museos con pocos recursos tanto económicos como técnicos, puedan desarrollar de una manera más sencilla sus aplicaciones.

El código fuente de este proyecto está disponible en el siguiente enlace:

https://drive.google.com/drive/folders/1P8sW2anUBObLy1hoY_aZGFzZlhd5MbjX?dmr=1&ec=wgc-drive-%5Bmodule%5D-goto

1.1 Motivación

En nuestra actualidad [1,2], la gran parte de los museos se enfrentan a adaptar sus colecciones a formatos tecnológicos, ya que ahora mismo la tecnología está en su auge y además, cada vez se van añadiendo tecnologías más innovadoras. Esto supone, que los museos tengan que integrarse con dichas tecnologías para poder llegar a más audiencia y diversa. Debido a las diferentes formas en las que las personas percibimos y disfrutamos de estas colecciones, hace necesario usar métodos más flexibles o adaptados a sus necesidades.

Es un hecho, que las generaciones futuras disfrutan más de colecciones culturales las cuales proporcionan tecnologías 3D o realidad aumentada, así mismo, hay una gran parte de personas que necesitan saber si los museos están adaptados a sus necesidades para poder disfrutar de ellos sin ninguna complicación. Esto provoca la necesidad de adaptarse a dichas actualizaciones, lo que supone que para muchos museos pueden

no tener estos recursos económicos y técnicos para poder adaptarse a dichas necesidades.

Por ejemplo, instituciones como el Museo del Louvre han introducido recorridos virtuales y aplicaciones móviles para ayudar y mejorar la experiencia de aquellos visitantes con movilidad reducida o diversidad sensorial. Dicho ejemplo muestra lo valioso que es tener herramientas que se puedan adaptar a distintas situaciones y ajustarse a lo que se necesita, pero es un gran problema para aquellos museos los cuales no tienen dichos recursos como museos tan importantes.

Este proyecto surge con el propósito de facilitar a los museos el desarrollo de sitios web personalizados reduciendo la necesidad de conocimientos técnicos avanzados. Se ha hecho a través de técnicas de desarrollo dirigidas por modelos y generación automática de código. Los museos podrán crear páginas web HTML o archivos JSON introduciendo únicamente los datos de sus colecciones. Este hecho les simplifica el proceso y reduce significativamente el esfuerzo técnico y económico requerido.

Además, la automatización del proceso permite una aproximación incremental, ya que los museos podrán empezar con soluciones básicas e ir agregando funcionalidades más avanzadas, como por ejemplo, modelos 3D, recorridos virtuales o realidad aumentada, en función de lo que les permita sus recursos.

Este proyecto se ha desarrollado utilizando las herramientas Xtext, Xtend y EMF en Eclipse, librerías de Python tales como BeautifulSoup y ElementTree para procesar los datos y generar los archivos XML y construir las respectivas bases de datos. La combinación de estas tecnologías ofrece una solución completa y automática, permitiendo a los museos crear sus sitios web a partir de datos estructurados, sin necesidad de conocimientos de programación avanzados una vez que la infraestructura necesaria está disponible.

De esta forma, el proyecto ayuda a los museos para que su contenido sea accesible y los disfrute todos los usuarios, promoviendo la inclusión y la innovación tecnológica en el sector museístico.

1.2 Objetivos

Este proyecto tiene como objetivo crear un sistema automatizado que permita a los museos generar archivos HTML y JSON, para poder crear y personalizar sus páginas webs, sin necesidad de que tengan altos conocimientos técnicos. El objetivo principal es facilitar y ayudar a la publicación de sus obras, y atraer a más usuarios para que disfruten e interactúan con ellas.

El sistema crea automáticamente los archivos con los datos añadidos tales como sus descripciones, imágenes, provincia, datos claves para ellos, permitiendo a los usuarios saber si un museo es adecuado o interesante para ellos antes de visitarlo. Además, se plantea incluir en el futuro recorridos virtuales, modelos 3D y realidad aumentada.

Los objetivos principales son:

- **La gestión de datos:** Organizar la información de los museos, como sus colecciones, materias y destacados, para poder trabajar con los datos estructurados correctamente y con la información necesaria.
- **Accesibilidad a los usuarios:** Crear páginas web claras, fáciles de comprender y adaptadas a los diferentes usuarios.
- **Facilidad de uso a los museos:** Permitir que los museos puedan publicar o añadir información de ellos, reduciendo su necesidad de conocimiento de la programación.
- **Interactividad y diseño:** Mostrar la información de manera atractiva y fácil de manejar para los diferentes usuarios.
- **Expansión futura:** Preparar el sistema para incorporar nuevas funciones como aplicaciones móviles y modelos 3D.

Este proyecto busca ayudar el acceso a la información cultural de las personas y mejorar sus experiencias en ellas, a la vez que ayuda a los museos a llegar a más personas y facilitar el desarrollo de sus aplicaciones.

1.3 Plan de trabajo

1. Investigación preliminar (1-2 meses): El objetivo de esta primera fase fue investigar acerca de los diferentes lenguajes en el Modeling Tools de Eclipse, su funcionamiento, complejidad y los diferentes resultados que proporcionan cada uno. Además, buscar las diferentes páginas web sobre museos, analizar los datos que proporcionaban cada una de ellas y la complejidad de su código.
2. Extraer la información de las páginas web (3 – 4 semanas): Esta segunda fase consistió en el aprendizaje de Web Scraping en Python, como en su desarrollo para extraer la información de las páginas web seleccionadas y almacenarlas en un XML.
3. Desarrollo con los lenguajes Xtext, Xtend y EMF (2 – 3 meses): Esta última fase consistió en el aprendizaje y desarrollo de los lenguajes Xtext, Xtend y EMF, en los que devuelven como solución páginas web en HTML y archivos JSON.

Capítulo 2 - Estado de la cuestión

2.1 Introducción

Las aplicaciones relacionadas con museos han ganado una gran importancia en los últimos años, ya que facilitan el acceso a los usuarios a su información cultural y además, mejoran la experiencia de ellos. Dichas aplicaciones abarcan desde simples exposiciones hasta complejas plataformas interactivas, las cuales incluyen recorridos virtuales, realidad aumentada y obras en 3D. Estos desarrollos han provocado diversos problemas a la hora de gestionar grandes cantidades de datos y conocimientos bastante avanzados de la programación, problemas a los la gran mayoría de museos se han enfrentado debido a que no tienen los mismos recursos económicos que otros museos, viéndose así afectados con las visitas de sus usuarios.

Uno de los principales retos es la estructuración y digitalización de datos, sobre todo para aquellos museos que tienen sus bases de datos fragmentadas o aún no han llegado a digitalizarlas [8]. Las innovaciones en la tecnología como la inteligencia artificial y la realidad aumentada mejoran la experiencia del usuario, por lo que los museos atraen a un público mayor, pero por otro lado añaden dificultades al desarrollo y económicamente de estas. Además, asegurar la accesibilidad de los diferentes usuarios debido a sus diversas capacidades implica implementar estándares como las Pautas de Accesibilidad para el Contenido Web (WCAG) y la compatibilidad con tecnologías de apoyo [9].

Como mencioné anteriormente, el costo de estas aplicaciones es bastante importante. Muchos museos dependen de soluciones de código abierto o de colaboraciones con universidades y empresas tecnológicas, lo que les limita a poder añadir funciones personalizadas o de mayor avance según sus deseos.

Para hacer frente a este problema, es necesario utilizar herramientas de desarrollo que simplifiquen el proceso de creación y mantenimiento. El uso de los lenguajes de modelado y lenguajes de automatización son muy útiles en estos aspectos, ya que sirven para desarrollar aplicaciones complejas, manejan grandes cantidades de datos y permiten personalizar las soluciones. Los lenguajes de modelado ayudan a representar

la idea de la solución y las relaciones de manera clara (diagramas o estructuras fáciles de entender), lo que mejora el entendimiento de las personas que trabajan en ella o quieren modificar partes del proyecto, este hecho se realiza de manera sencilla y reduce errores.

Por otro lado, los lenguajes de automatización permiten crear automáticamente código o interfaces, esto ahorra mucho tiempo y esfuerzo porque no es necesario realizar las funciones básicas a mano.

Elegí estos tipos de lenguajes ya que me parecieron herramientas flexibles, las cuales se pueden empezar con soluciones simples, y poco a poco, ir agregando funciones más avanzadas adaptándose a diferentes necesidades, por ejemplo añadir muchos más datos o modificar la idea principal que tenía de la solución.

Además de estos lenguajes, también se empleó la técnica de Web Scraping para recoger los datos de las páginas web relacionadas con los museos en internet, y convertirlos en datos estructurados personalizados, para poder generar archivos con los que trabajaré. Web Scraping es muy útil en contextos donde no existe una base de datos pública y organizada, como me ocurrió en mi caso.

2.2 Web Scraping

Como se ha explicado anteriormente, se utilizó esta técnica en Python para poder recopilar los datos seleccionados de las dos páginas webs escogidas, y poder trabajar con los archivos XML generados que contienen los datos estructurados. Las dos páginas web escogidas fueron: Europeana [10, 11], para obtener información sobre las materias y destacados, y el Directorio de Museos y Colecciones de España [12], para obtener información sobre los diferentes museos de España y su localización.

Entre las diferentes librerías que existen para realizar Web Scraping en Python, se seleccionaron Beautiful Soup y ElementTree debido a sus características específicas las cuales se ajustaban a las necesidades del proyecto.

Beautiful Soup: Es una librería que permite trabajar con el código HTML de las páginas webs y encontrar fácilmente los datos que se necesitan extraer de este código.

Element Tree: Es una librería que sirve para crear archivos XML y guardar los datos de forma estructurada.

2.3 Lenguajes de Modelado

Los lenguajes de modelado son herramientas que sirven para desarrollar soluciones complejos, ya que permiten definir, estructurar y visualizar los datos y procesos de manera abstracta. Su función principal es que representan ideas, conceptos y relaciones a través de diagramas y esquemas, los cuales facilitan la comprensión a las personas implicadas o aquellos que un futuro quieran modificar el proyecto.

En el contexto de este proyecto, los lenguajes de modelado se utilizaron para establecer una base sólida en la definición de datos y la generación automática de componentes clave. Las principales herramientas empleadas fueron:

Xtext: es un marco que permite definir lenguajes específicos de dominio (DSL) mediante gramáticas personalizadas. Como resultado, se obtiene una infraestructura completa, la cual incluye analizadores, enlazadores y código fuente basado en el modelo definido.

Eclipse Modeling Framework (EMF): Es un marco de modelado que permite definir modelos de datos detallados y generar automáticamente código Java para manejar estos modelos. EMF también soporta la creación de editores visuales y la persistencia de datos.

Ambos lenguajes han sido fundamentales para estructurar los datos de manera clara, relacionarlos y generar componentes, lo que ha reducido significativamente el tiempo de desarrollo y los posibles errores.

2.4 Lenguajes de Automatización

Los lenguajes de automatización son esenciales en el desarrollo de sistemas complejos, ya que permiten generar automáticamente diversos elementos como código fuente, estructuras de datos e interfaces de usuario. Su implementación ahorra tiempo, minimiza errores manuales y garantiza la coherencia en el desarrollo, especialmente en proyectos de gran tamaño.

En este proyecto, los lenguajes de automatización desempeñaron un papel crucial para transformar los modelos en implementaciones funcionales y reducir el esfuerzo de la programación manual. Las herramientas utilizadas incluyen:

Xtend: Un lenguaje de programación orientado a objetos que extiende de Java, ofreciendo una sintaxis más concisa y legible. Xtend se ha juntado con Xtext, para permitir transformar modelos en implementaciones concretas de manera clara y eficiente. Su capacidad para generar código simplifica la gestión de proyectos complejos.

Eclipse Modeling Framework (EMF): Además de su capacidad de modelado que mencionamos anteriormente, permite la generación automática de código Java para manejar los datos definidos en los modelos. Esto asegura que las aplicaciones desarrolladas sean coherentes, escalables y fáciles de mantener.

La combinación de estas tecnologías permite desarrollar un entorno automatizado y extensible, que genera aplicaciones personalizadas para museos adaptadas a las diferentes necesidades de los museos De esta manera, se garantiza una experiencia fácil para los dueños de los museos así como para futuros desarrolladores.

Capítulo 3 - Implementación de mi proyecto

En este capítulo se va a desarrollar todo el proceso del proyecto, como se obtuvo y se prepararon los datos de las páginas webs de museos escogidas. Y el principal objetivo que fue crear como solución los archivos HTML y JSON, con los datos obtenidos. Las páginas web que se seleccionaron fueron Europeana [10,11] y el Directorio de Museos y Colecciones de España [12]. Europeana me llamó la atención debido a que mostraba información cultural diferente al resto de páginas webs, y el Directorio de Museos y Colecciones de España, para poder ver en que municipios tenía cada museo y colección. Además, ambas tienen acceso público y una estructura bien definida, lo que me facilitó la extracción de datos.

Para ellos, se ha empleado la técnica de web scraping, que permite extraer de forma automatizada datos de páginas webs y guardarla en un archivo XML. Para el proceso de extracción de datos se utilizó la librería BeautifulSoup en Python, ya que permite analizar el código HTML de la página web y extraer a través de las etiquetas, los elementos que necesitaba. A continuación, con la librería ElementTree se transformaron y estructuraron dichos elementos para guardarlos en un archivo XML. Estos archivos XML han sido la base para poder crear la base de datos y generar las soluciones de HTML y JSON.

Además, se han utilizado los lenguajes Xtext y Xtend para la creación de un lenguaje especifico de dominio que me ha permitido relacionar la información en base a la representación que quería que tuviese la solución de HTML, además de implementar un archivo Xtend el cual contiene la estructura de dicha solución.

Y por último, se utilizó el lenguaje EMF donde se creó un metamodelo especificando las relaciones que iban a tener los datos de los museos en la solución JSON. Y también se implementó un archivo java, para poder leer el XML y estructurar y generar la solución.

Este capítulo presenta una descripción detallada de las herramientas y lenguajes empleadas para la extracción, transformación, almacenamiento y manipulación de los datos de museos, así como la generación de los diferentes archivos. Además, se explica

el porqué de cada herramienta y como se ha usado para poder lograr el objetivo final, el cual puede ser mejorado en futuras investigaciones.

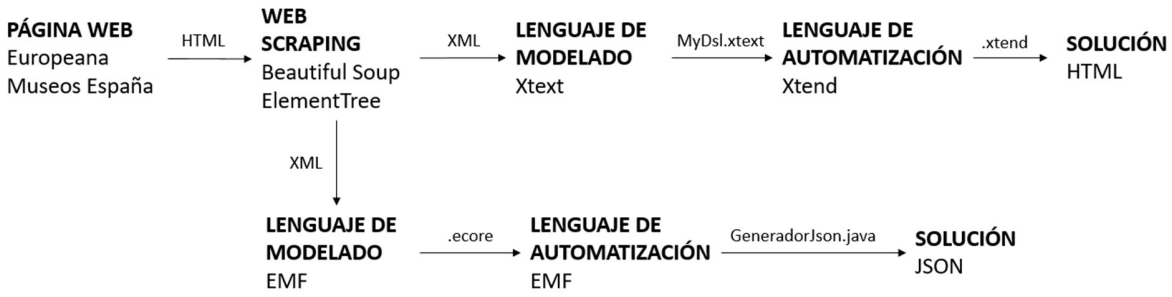


Figura 3-1 Estructura

3.1 Web Scraping en Python

¡La misma semana de la entrega del proyecto, la página web Europeana añadió que se identificase si al cargarse la página web era un bot o no, lo que finalmente al compilar mi archivo de materiasydestcados.ipynb diese error, aunque en la memoría tenga las capturas de este archivo, en el repositorio del proyecto, solo pude salvar un archivo que tenía donde había cargado las materias!

Para iniciar el proyecto, seleccioné dos páginas web que sirvieron como la base para el desarrollo del mismo: Europeana [10, 11] donde obtuve las materias y los destacados, y el Directorio de Museos y Colecciones de España [12] donde obtuve información correspondiente de diversos museos de España.

En primer lugar, elegí la página de Europeana, ya que al examinar su código fuente, encontré que su estructura era clara y sencilla, lo que facilitaba el uso y la implementación de las técnicas de Web Scraping utilizando Python. Posteriormente, cuando adquirí mayor experiencia en el manejo de Web Scraping, decidí incorporar también la página del Directorio de Museos, cuyo código era algo más complejo, pero considere que la información de la página era muy interesante para extraer datos sobre los diferentes museos de España.

Para programar en Python, utilice Anaconda Jupyter. Como mencioné anteriormente, empecé con la página web Europeana (materias y destacados) ya que

su código HTML se mostraba más claro y sencillo. En el código de Phyton, use la librería requests para hacer una solicitud URL a la página web y obtener así su código HTML. A continuación, utilicé el analizador BeautifulSoup [3] para hacer Web Scraping, ya que analiza la página web y luego busca las etiquetas en el HTML. Busqué los "divs" (etiquetas del HTML) con ciertos valores para poder extraer la información que me interesaba. Extraje cada título, descripción y fuente de la imagen, y las almacené en unas listas (una para materias y otra para destacados).

Para finalizar, utilicé la librería ElementTree para poder crear un archivo XML donde se guardase de forma estructurados los datos de las materias y los destacados. Primero, se crea un elemento raíz llamado "museos" que funcionará como el contenedor de la información que tendrá el archivo XML. A continuación, se crea la sección de "materias" y se recorre la lista de ella, y seguidamente, se realiza el mismo proceso con "destacados". Finalmente, se genera un árbol XML a partir de la estructura creada, donde dicho árbol es una estructura jerárquica de los elementos y subelementos que se añadieron, para así generan el archivo XML llamado "museos.xml", asegurándose de que este bien codificado con UTF-8.

```
In [40]: import requests
         from bs4 import BeautifulSoup
         from IPython.display import Image, display
         from io import BytesIO
         soup = BeautifulSoup(result.text, 'html.parser')
         main_items = soup.find_all('div', {'data-v-1e4bdaf2': True, 'data-qa': 'content card', 'class': 'card text-left content-card defa
         dataMaterias = []
         for main in main_items:
             titulo = main.find('a', class_='card-link').text.strip()
             texto = main.find('div', class_='card-text').text.strip()
             img = main.find('source')['srcset']
             img_url= img.split()[0]

             dataMaterias.append({'titulo': f'"{titulo}"', 'descripcion':  f'"{texto}"', 'image':  f'"{img_url}"'})
             print(titulo)
             print(texto)
             display(Image(url=img_url, width=300, height= 300))
```

Figura 3-2 BeautifulSoup Materias

```
In [41]: import requests
         from bs4 import BeautifulSoup
         from IPython.display import Image, display

         soup = BeautifulSoup(result2.text, 'html.parser')

         main_items = soup.find_all('div', {'data-v-40f431aa': True, 'data-qa': 'content card',
                                            'class': 'card text-left content-card null-card'})

         dataDestacados = []

         for main in main_items:

             titulo = main.find('a', class_='card-link').text.strip()
             descripcion = main.find('div', class_='card-text').text.strip()
             source_tag = main.find('source')
             img_url = source_tag['srcset'].split()[0]

             dataDestacados.append({
                 'titulo': f'"{titulo}"',
                 'descripcion': f'"{descripcion}"',
                 'image': f'"{img_url}"'
             })
             print(titulo)
             print(descripcion)
             display(Image(url=img_url, width=300, height= 300))
```

Figura 3-3 BeautifulSoup Destacados

```
In [44]: import xml.etree.ElementTree as ET

         root = ET.Element('museos')
         materias_element = ET.SubElement(root, 'materias')
         for entry in dataMaterias:
             main = ET.SubElement(materias_element, 'materia')
             titulo = ET.SubElement(main, 'titulo')
             titulo.text = entry['titulo']
             descripcion = ET.SubElement(main, 'descripcion')
             descripcion.text = entry['descripcion']
             image = ET.SubElement(main, 'image')
             image.text = entry['image']
         destacados_element = ET.SubElement(root, 'destacados')
         for entry in dataDestacados:
             main = ET.SubElement(destacados_element, 'destacado')
             titulo = ET.SubElement(main, 'titulo')
             titulo.text = entry['titulo']
             descripcion = ET.SubElement(main, 'descripcion')
             descripcion.text = entry['descripcion']
             image = ET.SubElement(main, 'image')
             image.text = entry['image']
```

```
In [45]: tree = ET.ElementTree(root)
         tree.write('museo.xml', encoding='utf-8', xml_declaration=True)
```

Figura 3-4 Element Tree Materias y Destacados

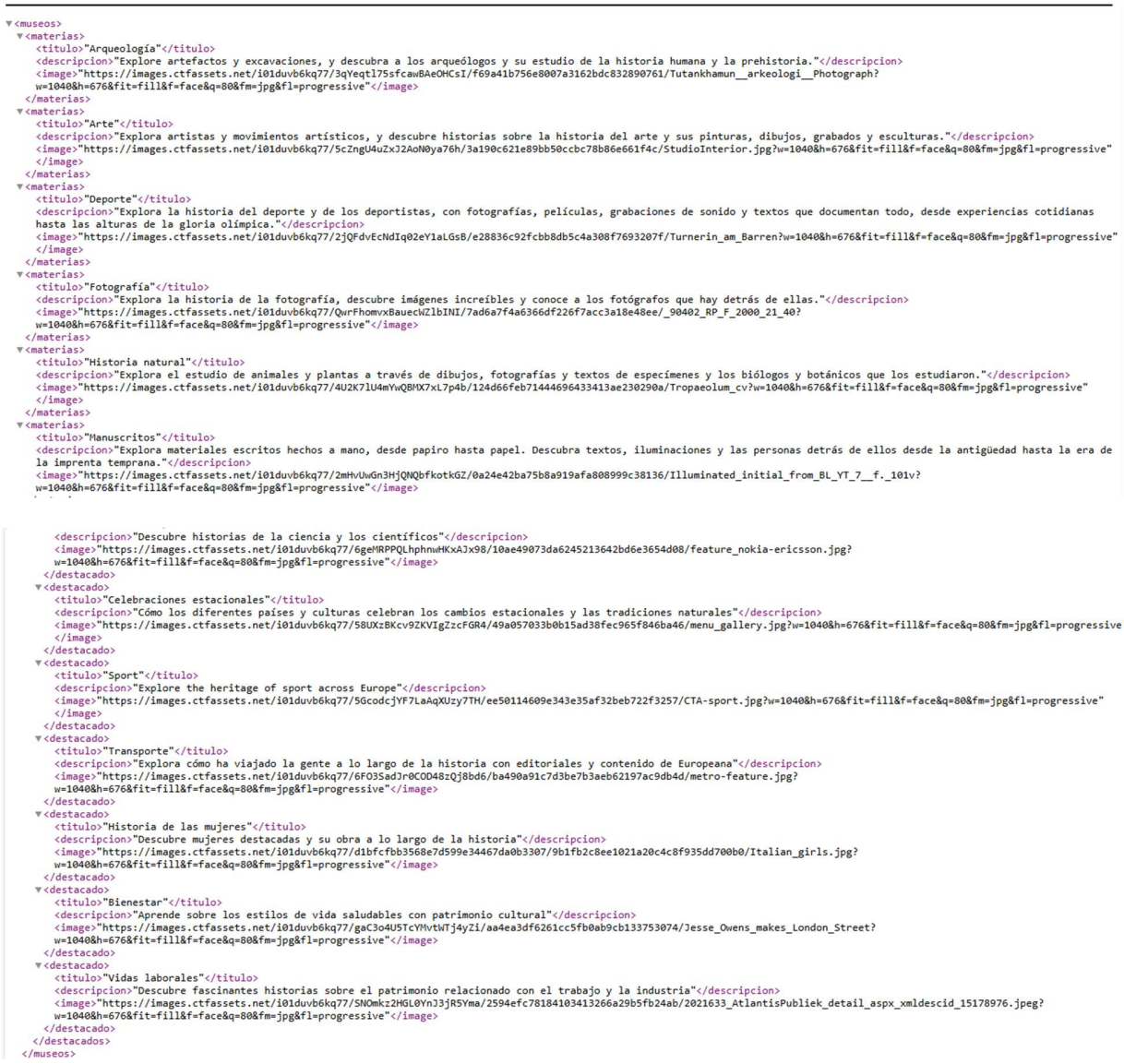


Figura 3-5 museo.xml

A continuación, conociendo mejor el Web Scraping, realicé el código del Directorio De Museos de España. El código realiza un scraping de datos sobre los museos. Envía solicitudes POST con parámetros de búsqueda generales, luego extrae la información de los museos (nombre, imagen, municipio, provincia y tipo) de la tabla HTML usando BeautifulSoup. Esta información se guarda de forma estructurada en un archivo XML utilizando ElementTree. El proceso incluye múltiples solicitudes y una pausa entre ellas para evitar sobrecargar del servidor. Estos datos se guardaron en el archivo museos.xml.

```python
import requests
from bs4 import BeautifulSoup
import time
import xml.etree.ElementTree as ET

urls = [
    "https://directoriomuseos.mcu.es/dirmuseos/realizarBusquedaGeneral.do",
    "https://directoriomuseos.mcu.es/dirmuseos/realizarBusquedaGeneral.do?tipoDeBusqueda=general&orderBy=nom&orderType=asc&pag=2"
]
payload = {
    "busqueda": "general",
    "accion": "Buscar",
    "provincia": "",
    "municipio": "",
    "tipo": "",
    "nombre": "",
}

headers = {
    "User-Agent":
    "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/114.0.0.0 Safari/537.36"
}

archivo = "museos.xml"
root = ET.Element("Museos")
MAX_PETICIONES = 3
# Para poder transformar las imagenes a URLs.
imagen_url = "https://directoriomuseos.mcu.es/dirmuseos/"
```

```python
for url in urls:
    for i in range(MAX_PETICIONES):
        response = requests.post(url, data=payload, headers=headers)
        soup = BeautifulSoup(response.text, "html.parser")
        rows = soup.find_all("tr")[1:]

        for row in rows:
            columns = row.find_all("td")
            if len(columns) >= 5:
                nombre = columns[0].text.strip()

                img = columns[1].find("img")
                imagen = img["src"] if img else "No disponible"
                if img and "src" in img.attrs:
                    src = img["src"]
                    if src != "images/default-destacada.JPG":
                        imagen = imagen_url + src

                municipio = columns[2].text.strip()
                provincia = columns[3].text.strip()
                tipo = columns[4].text.strip()

                museo = ET.SubElement(root, "Museo")

                ET.SubElement(museo, "Nombre").text = f'"{nombre}"'
                ET.SubElement(museo, "Imagen").text = f'"{imagen}"'
                ET.SubElement(museo, "Municipio").text = f'"{municipio}"'
                ET.SubElement(museo, "Provincia").text = f'"{provincia}"'
                ET.SubElement(museo, "Tipo").text = f'"{tipo}"'
```

Figura 3-6 BeautifulSoup Museos España

```python
                print(f"Museo: {nombre}, Imagen: {imagen}, Municipio: {municipio}, Provincia: {provincia}, Tipo: {tipo}")

        time.sleep(5)  # Para que no se sature el servidor


tree = ET.ElementTree(root)
tree.write(archivo, encoding="utf-8", xml_declaration=True)
```

Figura 3-7 ElementTree Museos España

Figura 3-8 museos.xml

3.2 Base de datos en SQL

Para esta fase del proyecto se creó una base de datos SQL a partir de los archivos XML para facilitar la gestión de los datos. Se usó la aplicación de Oracle SQL Developer.

Para comenzar, hay que crear una nueva conexión a la base de datos. Para ello, el SID tiene que cambiarse a orcl, y añadir el usuario y contraseña con el que se instaló la aplicación.

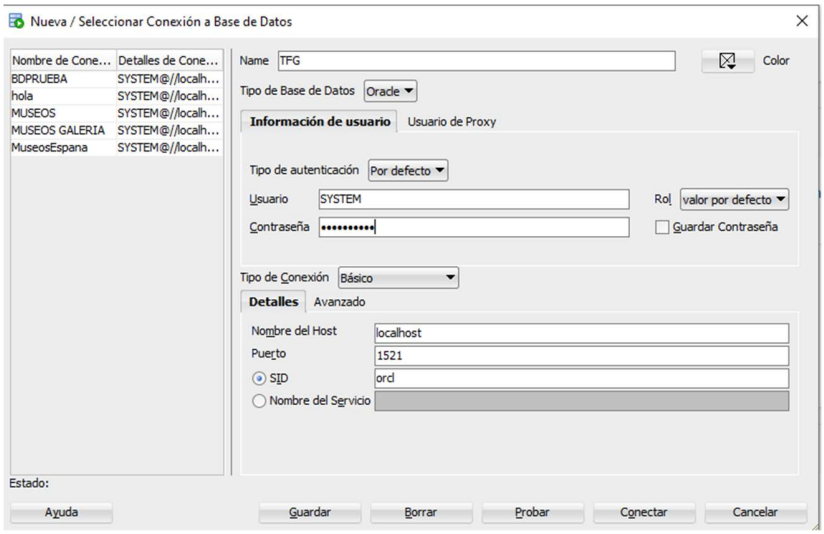


Figura 3-9 Conexión base de datos

En esta parte, empecé trabajando con la página web de Europeana, creando primero la estructura de la base de datos con sus respectivas tablas de materias y destacados, para luego insertar los datos manualmente usando el comando INSERT INTO. Sin embargo, me di cuenta de que este proceso era demasiado tedioso debido a la necesidad de añadir los datos uno por uno. Tras investigar, encontré una forma más eficiente de cargar los datos desde un archivo XML.

Para ello, creé las tablas materias y destacados con los campos necesarios, como título, descripción (usando el tipo de dato CLOB), e imagen. Luego, definí un directorio en el sistema que apuntaba al archivo "museo.xml", que contenía los datos a insertar. Para leer este archivo y convertirlo en un tipo de dato procesable, utilicé el procedimiento DBMS_XSLPROCESSOR.READ2CLOB, que lee el contenido del archivo "museo.xml" y lo almacena en una variable CLOB, ya que esta sirve para manejar grandes cantidades de texto.

A continuación, utilicé las funciones XMLTYPE y XMLSEQUENCE para procesar los datos. La función de XMLTYPE convierte el contenido de la variable CLOB en un formato para que este pueda ser analizado y consultado por otras funciones de XML Oracle. Por otro lado, XMLSEQUENCE genera una secuencia de todos los elementos que hay en la variable v_clob. Además, EXTRACTVALUE o EXTRACT, extrae los datos específicos. Con estas funciones, extraje los valores relevantes del archivo XML y los inserté directamente

en las tablas correspondientes de la base de datos. Finalmente, realicé un COMMIT para guardar de forma definitiva los cambios en la base de datos.

De esta manera, se automatiza la inserción de los datos desde el archivo "museo.xml", lo que simplifica enormemente el proceso.

```
CREATE TABLE materias (
    titulo VARCHAR2(255),
    descripcion CLOB,
    image VARCHAR2(255)
);
CREATE TABLE destacados (
    titulo VARCHAR2(255),
    descripcion CLOB,
    image VARCHAR2(255)
);
CREATE OR REPLACE DIRECTORY DIRECTORY_XML2 AS 'C:\Users\Paula\Desktop\AATFG';

GRANT READ, WRITE ON DIRECTORY DIRECTORY_XML2 TO SYS;
DECLARE
    v_clob CLOB;
BEGIN
    -- Leemos el archivo XML
    v_clob := DBMS_XSLPROCESSOR.READ2CLOB('DIRECTORY_XML', 'museo.xml');

    -- Insertamos los datos en la tabla 'materias'
    INSERT INTO materias (titulo, descripcion, image)
    SELECT
        EXTRACTVALUE(VALUE(x), '/materia/titulo'),
        EXTRACTVALUE(VALUE(x), '/materia/descripcion'),
        EXTRACTVALUE(VALUE(x), '/materia/image')
    -- Extraemos y convertimos los elementos <Materia> de v_clob en filas
    -- XMLSEQUENCE genera una secuencia de elementos
    -- EXTRACT para obtener los datos específicos
    FROM TABLE(XMLSEQUENCE(EXTRACT(XMLTYPE(v_clob), '/museo/materias/materia'))) x;
```

```
    -- Insertamos datos en la tabla 'destacados'
    INSERT INTO destacados (titulo, descripcion, image)
    SELECT
        EXTRACTVALUE(VALUE(x), '/destacado/titulo'),
        EXTRACTVALUE(VALUE(x), '/destacado/descripcion'),
        EXTRACTVALUE(VALUE(x), '/destacado/image')
    -- Extraemos y convertimos los elementos <Destacado> de v_clob en filas
    -- XMLSEQUENCE genera una secuencia de elementos
    -- EXTRACT para obtener los datos específico
    FROM TABLE(XMLSEQUENCE(EXTRACT(XMLTYPE(v_clob), '/museo/destacados/destacado'))) x;

    COMMIT;
END;
```

Figura 3-10 MateriasYDestacados.sql

	TITULO	DESCRIPCION	IMAGE
1	"Arqueología"	"Explore artefactos y excavaciones, y descubra a los arqueólogos y su es...	"https://images.ctfassets.net/i01duvb6kq77/3qYeqtl
2	"Arte"	"Explora artistas y movimientos artísticos, y descubre historias sobre l...	"https://images.ctfassets.net/i01duvb6kq77/5cZngU4
3	"Deporte"	"Explora la historia del deporte y de los deportistas, con fotografías, ...	"https://images.ctfassets.net/i01duvb6kq77/2jQFdvE
4	"Fotografía"	"Explora la historia de la fotografía, descubre imágenes increíbles y co...	"https://images.ctfassets.net/i01duvb6kq77/QwrFhom
5	"Historia natural"	"Explora el estudio de animales y plantas a través de dibujos, fotografí...	"https://images.ctfassets.net/i01duvb6kq77/4U2K71U
6	"Manuscritos"	"Explora materiales escritos hechos a mano, desde papiro hasta papel. De...	"https://images.ctfassets.net/i01duvb6kq77/2mHvUwG
7	"Mapas y geografía"	"Explora las características físicas y humanas de la Tierra. Descubre ma...	"https://images.ctfassets.net/i01duvb6kq77/6g2HPP0
8	"Migración"	"Los seres humanos siempre se han desplazado de un lugar a otro. Descubr...	"https://images.ctfassets.net/i01duvb6kq77/6rvABmI
9	"Moda"	"Explore la moda histórica y contemporánea y cómo influye en nuestra rop...	"https://images.ctfassets.net/i01duvb6kq77/2bQMPeI
10	"Música"	"Explora grabaciones, partituras, instrumentos y estilos musicales, así ...	"https://images.ctfassets.net/i01duvb6kq77/1uf2cle
11	"Patrimonio industrial"	"Explore la historia de la tecnología y la industria, desde la revolució...	"https://images.ctfassets.net/i01duvb6kq77/7GolTr4
12	"Periódicos"	"Explora publicaciones impresas desde 1618 hasta la década de 1980. Desc...	"https://images.ctfassets.net/i01duvb6kq77/44f3kvm
13	"Primera Guerra Mundial"	"Explore los eventos de 1914-1918 a través de historias oficiales, polít...	"https://images.ctfassets.net/i01duvb6kq77/2Q82YtI

1	"Siglo XX"	"Explora colecciones e historias de la historia y la cultura del siglo XX"	"https://images.ctfassets.net/i01duvb6kq77/
2	"Animales"	"Mejora tu conocimiento del mundo animal, la historia y la cultura"	"https://images.ctfassets.net/i01duvb6kq77/
3	"Art Nouveau"	"Explora el extenso archivo de Europeana de historias de Art Nouveau"	"https://images.ctfassets.net/i01duvb6kq77/
4	"Arquitectura"	"Descubre las maravillas de la arquitectura europea"	"https://images.ctfassets.net/i01duvb6kq77/
5	"Asian art & heritage"	"Explore stories and objects from Asia in European collections"	"https://images.ctfassets.net/i01duvb6kq77/
6	"Historia negra"	"Explora las historias de las personas negras en Europa"	"https://images.ctfassets.net/i01duvb6kq77/
7	"Broadcasting Europe"	"Explore Europe's audiovisual heritage"	"https://images.ctfassets.net/i01duvb6kq77/
8	"Chinese heritage"	"Explore stories and objects from China's rich history in European colle...	"https://images.ctfassets.net/i01duvb6kq77/
9	"Libros para colorear"	"Revitaliza la cultura con la gama de libros para colorear de Europeana"	"https://images.ctfassets.net/i01duvb6kq77/
10	"Colores"	"Llena tu vida de color con estos colores seleccionados"	"https://images.ctfassets.net/i01duvb6kq77/
11	"Compositores"	"Conoce a las personas que dieron forma a la historia y la cultura music...	"https://images.ctfassets.net/i01duvb6kq77/
12	"Patrimonio de la discapacidad"	"Explora historias de personas con discapacidad de toda Europa"	"https://images.ctfassets.net/i01duvb6kq77/
13	"Descubrir Europa"	"Haz un viaje virtual por Europa desde la comodidad de tu hogar"	"https://images.ctfassets.net/i01duvb6kq77/
14	"Diversidad e inclusión"	"Encuentra recursos para aprender y enseñar inclusión y diversidad"	"https://images.ctfassets.net/i01duvb6kq77/
15	"Dragones, mitos y leyendas"	"Aprende más sobre las criaturas legendarias de Europa"	"https://images.ctfassets.net/i01duvb6kq77/
16	"Entorno"	"Historias de Europeana sobre el entorno, el clima y la naturaleza."	"https://images.ctfassets.net/i01duvb6kq77/
17	"European identities"	"Discover curated content about European identities, cultures and history"	"https://images.ctfassets.net/i01duvb6kq77/
18	"Alimentos y bebidas"	"Explora el patrimonio cultural europeo a través de los alimentos y las ...	"https://images.ctfassets.net/i01duvb6kq77/
19	"Islamic heritage"	"Collections and stories about Islamic customs, tradition, culture and h...	"https://images.ctfassets.net/i01duvb6kq77/
20	"Patrimonio judío"	"Explora las costumbres, la tradición, la cultura y la historia judías"	"https://images.ctfassets.net/i01duvb6kq77/

Figura 3-11 Base de datos de Materias y Destacados

Posteriormente, procedí a crear la base de datos del Directorio de Museos de España utilizando el mismo procedimiento.

```
CREATE TABLE MuseosEspana (
    Nombre VARCHAR2(255),
    Imagen VARCHAR2(255),
    Municipio VARCHAR2(255),
    Provincia VARCHAR2(255),
    Tipo VARCHAR2(255)
);
CREATE OR REPLACE DIRECTORY DIRECTORY_XML AS 'C:\Users\Paula\Desktop\AATFG\';

GRANT READ, WRITE ON DIRECTORY DIRECTORY_XML TO SYS;
DECLARE
    v_clob CLOB;
BEGIN
    -- Leemos el archivo XML y lo almacenamos en v_clob
    v_clob := DBMS_XSLPROCESSOR.READ2CLOB('DIRECTORY_XML', 'museos.xml');

    -- Insertamos datos en la tabla
    INSERT INTO MuseosEspana (Nombre, Imagen, Municipio, Provincia, Tipo)
    SELECT
        EXTRACTVALUE(VALUE(x), '/Museo/Nombre'),
        EXTRACTVALUE(VALUE(x), '/Museo/Imagen'),
        EXTRACTVALUE(VALUE(x), '/Museo/Municipio'),
        EXTRACTVALUE(VALUE(x), '/Museo/Provincia'),
        EXTRACTVALUE(VALUE(x), '/Museo/Tipo')
    -- Extraemos y convertimos los elementos <Museo> de v_clob en filas
    -- XMLSEQUENCE genera una secuencia de elementos
```

```
    -- EXTRACT para obtener los datos especifico
    FROM TABLE(XMLSEQUENCE(EXTRACT(XMLTYPE(v_clob), '/Museos/Museo'))) x;
    COMMIT;
END;
```

Figura 3-12 MuseosEspana.sql

50	"ANTZASTI"	"images/default-destacada.JPG"	"Dima"
51	"AQUARIUM DONOSTIA-SAN SEBASTIAN"	"https://directoriomuseos.mcu.es/dirmuseos/descargarFichero.do?fileName=...	"Donostia-San S
52	"ARCHIVO - MUSEO IGNACIO SÁNCHEZ MEJÍAS"	"images/default-destacada.JPG"	"Manzanares"
53	"ARCHIVO MANUEL DE FALLA"	"images/default-destacada.JPG"	"Granada"
54	"ARCHIVO MUSEO "DON ÁLVARO DE BAZÁN""	"images/default-destacada.JPG"	"Viso del Marqu
55	"ART MAJOR DE LA SEDA"	"https://directoriomuseos.mcu.es/dirmuseos/descargarFichero.do?fileName=...	"Valencia"
56	"ARTIUM MUSEOA. MUSEO DE ARTE CONTEMPORÁNEO DEL PAÍS VASCO"	"images/default-destacada.JPG"	"Vitoria-Gastei
57	"ARXIU I MUSEU DE L'EDUCACIÓ DE LES ILLES BALEARS"	"https://directoriomuseos.mcu.es/dirmuseos/descargarFichero.do?fileName=...	"Inca - Mallorc
58	"ASOCIACION CULTURAL SANTISIMO CRISTO DE LA VERA CRUZ"	"images/default-destacada.JPG"	"Consuegra"
59	"ASOCIACIÓN DE ESTUDIOS MELILLENSES"	"images/default-destacada.JPG"	"Melilla"
60	"AULA DIDÁCTICA DEL CASTRO DE COAÑA"	"https://directoriomuseos.mcu.es/dirmuseos/descargarFichero.do?fileName=...	"Coaña"
61	"AULA MUSEO PACO DÍEZ"	"images/default-destacada.JPG"	"Mucientes"
62	"BARCO MUSEO MATER. CENTRO DE INTERPRETACIÓN DE LA PESCA"	"images/default-destacada.JPG"	"Pasaia"
63	"BASÍLICA DE LA SANTA CRUZ DEL VALLE DE LOS CAÍDOS"	"images/default-destacada.JPG"	"San Lorenzo de
64	"BATÁN-MUSEO Y C.I. TEXTIL "LA COMUNAL""	"images/default-destacada.JPG"	"Val de San Lor
65	"BIBAT. MUSEO DE ARQUEOLOGÍA Y MUSEO FOURNIER DE NAIPES"	"https://directoriomuseos.mcu.es/dirmuseos/descargarFichero.do?fileName=...	"Vitoria-Gastei
66	"BIBLIOTECA-MUSEU VÍCTOR BALAGUER"	"images/default-destacada.JPG"	"Vilanova i la
67	"CA N'OLIVER. COL·LECCIÓ HERNÁNDEZ SANZ - HERNÁNDEZ MORA"	"images/default-destacada.JPG"	"Mahón - Menorc
68	"CAIXAFORUM PALMA"	"images/default-destacada.JPG"	"Palma Mallorca
69	"CALCOGRAFÍA NACIONAL"	"images/default-destacada.JPG"	"Madrid"
70	"CAN BALAGUER"	"images/default-destacada.JPG"	"Palma Mallorca
71	"CAN QUINTANA. CENTRE CULTURAL DE LA MEDITERRÀNIA"	"images/default-destacada.JPG"	"Torroella de M
72	"CASA - MUSEO ANTONIO PADRÓN"	"images/default-destacada.JPG"	"Gáldar"
73	"CASA - MUSEO LEÓN Y CASTILLO"	"images/default-destacada.JPG"	"Telde"
74	"CASA - MUSEO PÉREZ GALDÓS"	"images/default-destacada.JPG"	"Palmas de Gran
75	"CASA - MUSEO UNAMUNO"	"images/default-destacada.JPG"	"Puerto del Ros

Figura 3-13 Base de datos Museos España

3.3 Lenguajes Xtext y Xtend.

3.3.1 Introducción

A continuación, se ha utilizado Xtext y Xtend en Eclipse para desarrollar una solución que permite generar un archivo HTML a partir de los archivos XML ya creados. Ésta se basa en la creación de un Lenguaje Específico de Dominio (DSL) para modelar los museos, utilizando Xtext para la definición de la gramática y Xtend para la lógica de generación del HTML.

En este proyecto, Xtext [4, 5] se ha utilizado para definir un DSL que describe museos, incluyendo secciones de "materias" y "destacados". Dicho DSL utiliza una sintaxis basada en etiquetas XML, lo que permite representar de manera estructurada la información relevante de cada museo.

Por otro lado, Xtend [6] es un lenguaje de programación que se integra con Xtext y se utiliza para implementar la lógica del generador de código. Xtend se utiliza aquí para escribir el generador que toma como entrada un archivo XML y produce un archivo HTML. Xtend permite manipular de manera sencilla las estructuras de datos y generar dinámicamente contenido HTML.

3.3.2 Descripción del Código y su Flujo de Ejecución

3.3.2.1 Materias y Destacado

El proceso comienza con la definición del lenguaje en el archivo MyDsl.xtext. En este archivo, se ha establecido la gramática del DSL, que describe la estructura de un museo y sus secciones. La gramática define dos secciones principales dentro de un museo: materias y destacados. Ambas contienen múltiples elementos, los cuales tienen tres atributos: titulo, descripción e image. Este hecho permite que cada museo tenga una serie de materias y destacados, cada uno con su título, descripción e imagen asociada. La gramática también establece que los elementos materias y destacados se deben incluir dentro de un contenedor principal <museos>.

```
1  grammar org.xtext.example.mydsl1.MyDsl with org.eclipse.xtext.common.Terminals
2
3  generate myDsl "http://www.xtext.org/example/mydsl1/MyDsl"
4
5  Model:
6      '<museos>'
7      '<materias>'sections+=Materias*'</materias>'
8      '<destacados>'sections+=Destacados*'</destacados>'
9      '</museos>';
10
11 Materias:
12     '<materia>'
13         '<titulo>' titulo=STRING '</titulo>'
14         '<descripcion>' descripcion=STRING '</descripcion>'
15         '<image>' image=STRING '</image>'
16     '</materia>';
17
18 Destacados:
19     '<destacado>'
20         '<titulo>' titulo=STRING '</titulo>'
21         '<descripcion>' descripcion=STRING '</descripcion>'
22         '<image>' image=STRING '</image>'
23     '</destacado>';
```

Figura 3-14 MyDsl.xtext Materias y Destacados

El archivo "MyDslGenerator.xtend" genera dinámicamente la página HTML a partir del archivo "museo.xml" el cual como hemos comentado anteriormente tiene la

misma estructura definida que “MyDsl.xtext. Transforma los datos almacenados en el archivo “museo.xml” en una representación visual que es consultada a través de un navegador web. La generación del HTML se realiza de manera automática y estructurada, lo que facilita la creación de páginas dinámicas sin necesidad de escribir el código HTML a mano.

El código empieza con la creación de la estructura básica del archivo HTML, Genera las etiquetas estándar de un archivo HTML: las etiquetas `<html>`, `<head>` y `<body>`. Además, se añade en el encabezado código para el CSS, que define el estilo de la página y un pequeño script en JavaScript que agrega interactividad a la página. Este script permite que, al hacer clic en un botón, se muestre o se oculte la información asociada a cada elemento de materias y destacados.

Luego se carga el archivo “museo.xml” que contiene la información de las materias y destacados. A continuación, el generador recorre el modelo que hemos cargado, donde contiene los elementos “materias” y “destacados”. Para cada uno de estos dos elementos, el código genera el bloque de HTML correspondiente, donde para cada materia o destacado, se crea un botón con el título y el contenido que incluye la descripción y la imagen asociada. Este bloque HTML también incluye el código necesario para que, al hacer clic en el botón, la información se muestre u oculte dinámicamente, proporcionando una experiencia interactiva al usuario.

```
resource.allContents.forEachRemaining [ e |
    if (e instanceof Model) {
        val model = e as Model


        html.append("<h2>Materias</h2>")
        var index = 0
        for (materias : model.sections) {
            if (materias instanceof Materias) {
                val sectionId = "materia_" + index
                html.append("<button class='button' onclick='toggleSection(\"").append(sectionId).append("\")'>")
                html.append(materias.titulo).append("</button>")
                html.append("<div class='section' id='").append(sectionId).append("'>")
                html.append("<p>").append(materias.descripcion).append("</p>")
                html.append("<img src='").append(materias.image).append("' alt='")
                    .append(materias.titulo).append("' width='400px'>")
                html.append("</div>")
                index = index + 1
            }
        }


        html.append("<h2>Destacados</h2>")
        var indexDestacado = 0
        for (destacado : model.sections) {
            if (destacado instanceof Destacados) {
                val sectionId = "destacado_" + indexDestacado
                html.append("<button class='button' onclick='toggleSection(\"").append(sectionId).append("\")'>")
                html.append(destacado.titulo).append("</button>")
                html.append("<div class='section' id='").append(sectionId).append("'>")
                html.append("<p>").append(destacado.descripcion).append("</p>")
                html.append("<img src='").append(destacado.image).append("' alt='")
                    .append(destacado.titulo).append("' width='400px'>")
                html.append("</div>")
                indexDestacado = indexDestacado + 1
            }
```

Figura 3-15 MyDslGenerator.xtend Materias y Destacados

Finalmente para generar el archivo "museum_sections.html", se ejecuta la carpeta "org.xtext.example.mydsl1" como Eclipse Application. Una vez abierto, hay que asegurarse de que el workspace donde esta es "runtime-EclipseApplication", después se creará una carpeta y dentro de "src", un archivo el cual tenga de extensión ".mydsl1". Al crear este archivo, se enlazará con nuestro proyecto y se deberán copiar los datos de "museo.xml" eliminando el encabezado para generar el HTML, el cual estará ubicado en la subcarpeta "src-gen"

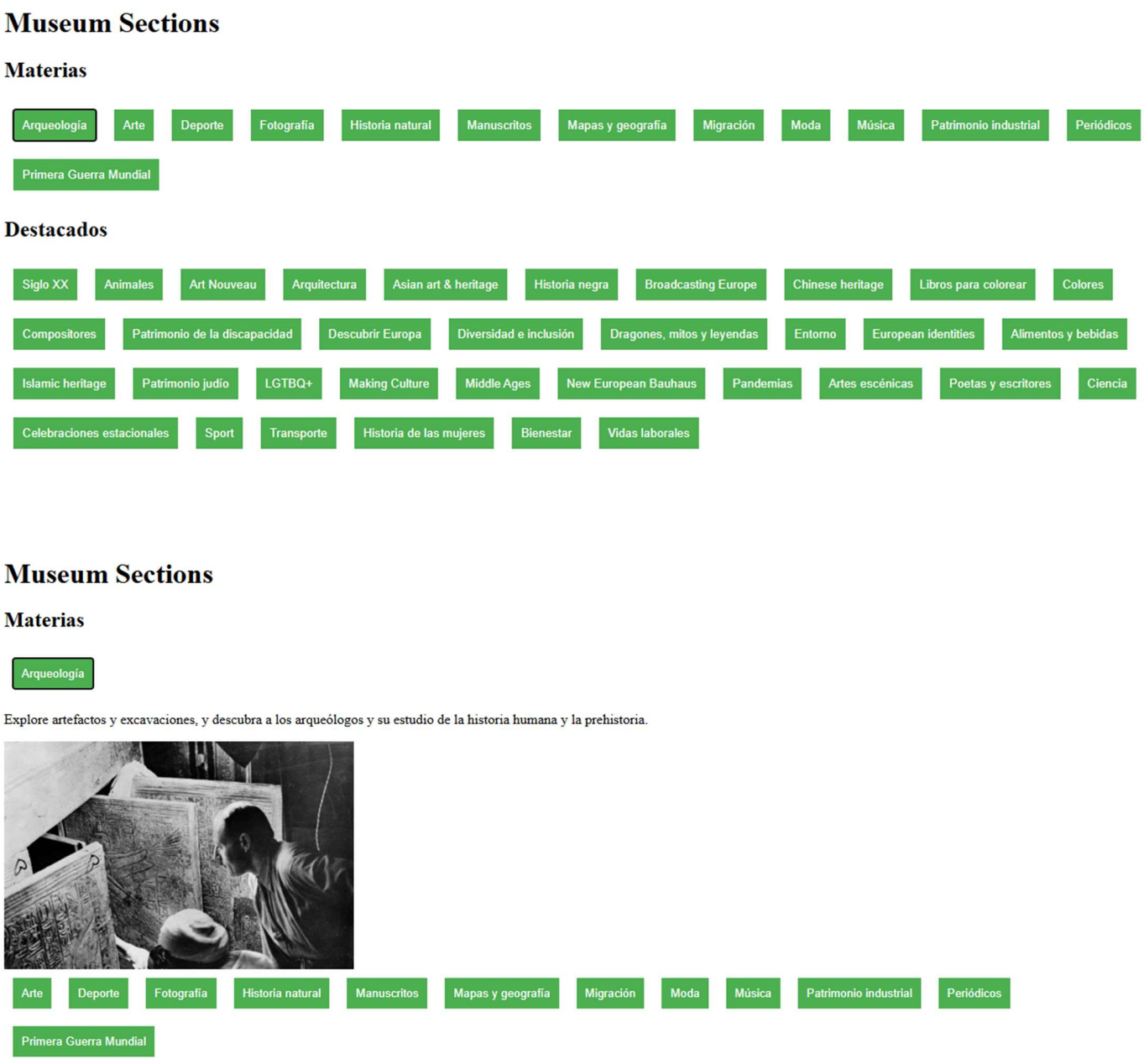


Figura 3-16 museum_sections.html

3.3.2.2 Museos España

A continuación, empecé con la página web de Museos España. El archivo MyDsl.xtext describe la estructura del museo, que incluye su nombre, imagen, municipio, provincia y tipo. Estos elementos se agrupan en una sección principal denominada <Museos> , la cual puede contener múltiples instancias de <Museo> . La gramática

establece claramente cómo se deben organizar estos elementos dentro del archivo, permitiendo que cada museo tenga su propio conjunto de atributos.

```
1  grammar org.xtext.example.mydsl.MyDsl with org.eclipse.xtext.common.Terminals
2
3  generate myDsl "http://www.xtext.org/example/mydsl/MyDsl"
4
5  Model:
6      '<Museos>'
7      sections+=Museo*
8      '</Museos>';
9
10 Museo:
11     '<Museo>'
12         '<Nombre>' nombre=STRING '</Nombre>'
13         '<Imagen>' imagen=STRING '</Imagen>'
14         '<Municipio>' municipio=STRING '</Municipio>'
15         '<Provincia>' provincia=STRING '</Provincia>'
16         '<Tipo>' tipo=STRING '</Tipo>''</Museo>' ;
17
```

Figura 3-17 MyDsl.xtext Museos España

En el archivo "MyDslGenerator.xtend", el código comienza con la creación de la estructura básica del archivo HTML, usa una variable llamada StringBuilder para generar dinámicamente el contenido del archivo HTML. Primero, se agregan las etiquetas estándar de un documento HTML, como <html>, <head>, y <body>. Dentro de la etiqueta <head>, se inserta código para el CSS y un bloque de JavaScript que como mencionamos anteriormente proporciona interactividad.

El bloque de CSS define el estilo para las tablas y los botones, para que las celdas de la tabla tengan bordes y espacio, y que los encabezados de la tabla tengan un fondo gris claro, para poder resaltar las características que hay. Los botones también se cambian de color y además se les agrega un efecto visual para que cuando el usuario pase el cursor sobre ellos, cambien su color del fondo a verde.

El bloque de JavaScript incluye dos funciones que permiten la interactividad en la página. La función showMuseos() muestra solo las filas correspondientes a los museos y oculta las de las colecciones, mientras que la función showColecciones() hace lo contrario, mostrando las colecciones y ocultando los museos. Esto se logra utilizando el método querySelectorAll() para seleccionar las filas correspondientes en la tabla y luego modificar su estilo de visualización.

Al final el código carga el archivo "museos.xml", el cual contiene la información sobre los museos y las colecciones, y sigue la estructura definida en "MyDsl.xtext", como mencionamos anteriormente. El código recorre el contenido de este archivo usando el método resource.allContents.forEachRemaining. A continuación, se comprueba si el elemento es una instancia de "Model" y, si lo es, se procesan las secciones del modelo y luego se filtran las secciones del modelo para obtener solo los museos y las colecciones, lo que permite separar ambos tipos de elementos. Los museos y las colecciones se almacenan en dos listas diferentes, "museos" y "colecciones", que más tarde se utilizarán para generar el contenido de la tabla.

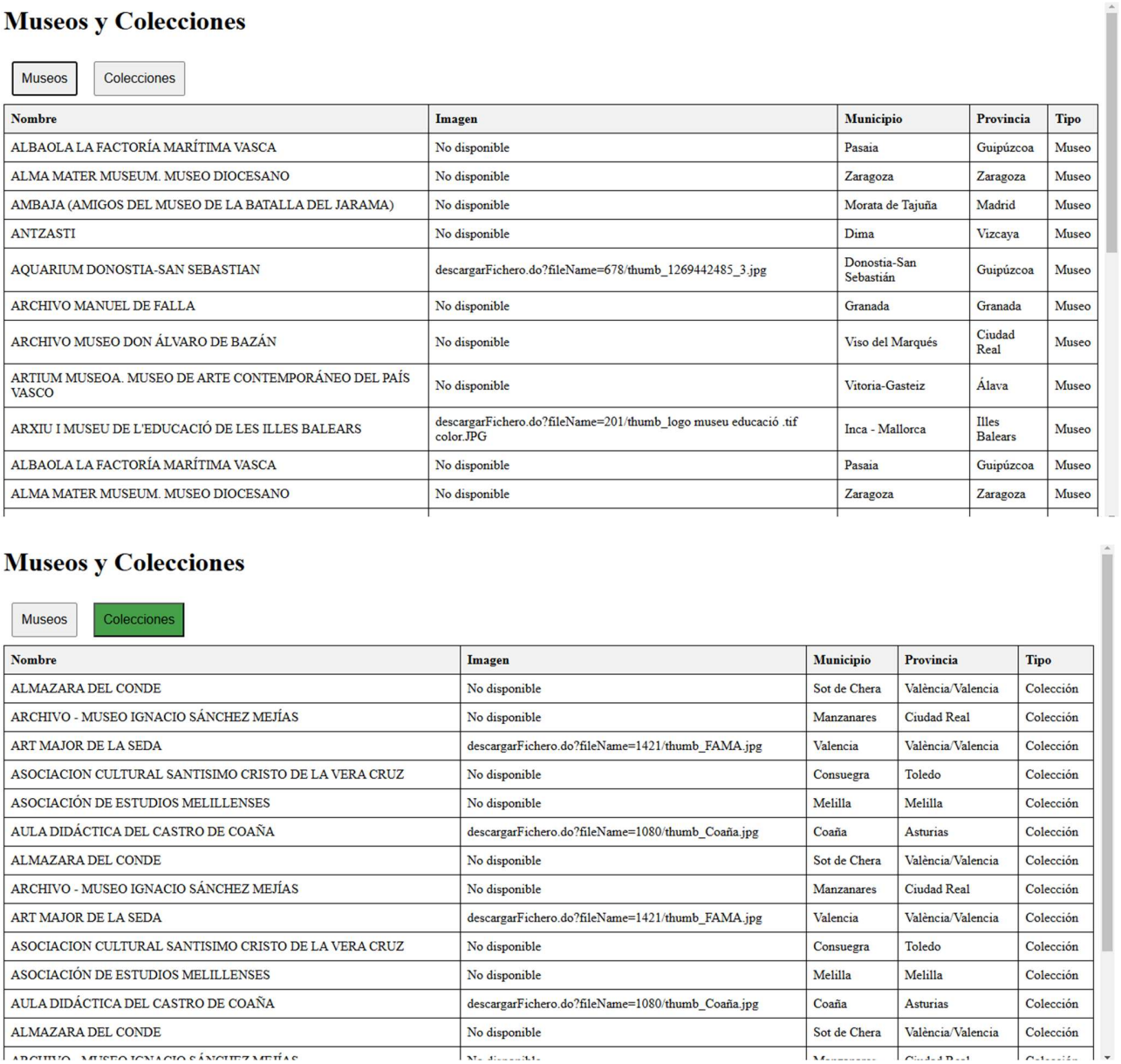
A continuación, el código recorre ambas listas, y genera dinámicamente una fila de tabla para cada elemento y agrega celdas con los datos, como el nombre, la imagen, el municipio, la provincia y el tipo. Estas celdas se agregan al archivo HTML utilizando el método html.append(). Cada fila de la tabla tiene una clase que se utiliza para identificarla y cambiar su visibilidad mediante el bloque de JavaScript.

El archivo generado se guarda como "museos_y_colecciones.html" en la carpeta correspondiente utilizando el método fsa.generateFile("museos_y_colecciones.html", html.toString()).

```
56
57⊖        resource.allContents.forEachRemaining [ e |
58             if (e instanceof Model) {
59                 val model = e as Model
60                 val museos = model.sections.filter[m | m.tipo == 'Museo']
61                 val colecciones = model.sections.filter[m | m.tipo == 'Colección']
62 System.out.println("Model: " + colecciones)
63⊖model.sections.forEach [ c |
64     System.out.println("Museo: " + c.nombre + " Tipo: " + c.tipo)
65 ]
66
67⊖                 museos.forEach [ museo |
68                     html.append("<tr class='Museo'>")
69                     html.append("<td>").append(museo.nombre).append("</td>")
70                     html.append("<td>").append(museo.imagen).append("</td>")
71                     html.append("<td>").append(museo.municipio).append("</td>")
72                     html.append("<td>").append(museo.provincia).append("</td>")
73                     html.append("<td>").append(museo.tipo).append("</td>")
74                     html.append("</tr>")
75                 ]
76
77
78⊖                 colecciones.forEach [ coleccion |
79                     html.append("<tr class='coleccion'>")
80                     html.append("<td>").append(coleccion.nombre).append("</td>")
81                     html.append("<td>").append(coleccion.imagen).append("</td>")
82                     html.append("<td>").append(coleccion.municipio).append("</td>")
83                     html.append("<td>").append(coleccion.provincia).append("</td>")
84                     html.append("<td>").append(coleccion.tipo).append("</td>")
85                     html.append("</tr>")
86                 ]
87             }
88         ]
89
```

Figura 3-18 MyDslGenerator.xtend Museos España

Finalmente, para generar el archivo "museos_y_colecciones.html" se ejecuta la carpeta "org.xtext.example.mydsl" como Eclipse Application. Una vez abierto, hay que asegurarse de que el workspace donde esta es "runtime-EclipseApplication", después se creará una carpeta y dentro de "src", un archivo el cual tenga de extensión ".mydsl". Al crear este archivo, se enlazará con nuestro proyecto y se deberán de copiar los datos de "museos.xml" eliminando el encabezado para generar el HTML, el cual estará ubicado en la subcarpeta "src-gen".



Museos y Colecciones

Museos | Colecciones

Nombre	Imagen	Municipio	Provincia	Tipo
ALBAOLA LA FACTORÍA MARÍTIMA VASCA	No disponible	Pasaia	Guipúzcoa	Museo
ALMA MATER MUSEUM. MUSEO DIOCESANO	No disponible	Zaragoza	Zaragoza	Museo
AMBAJA (AMIGOS DEL MUSEO DE LA BATALLA DEL JARAMA)	No disponible	Morata de Tajuña	Madrid	Museo
ANTZASTI	No disponible	Dima	Vizcaya	Museo
AQUARIUM DONOSTIA-SAN SEBASTIAN	descargarFichero.do?fileName=678/thumb_1269442485_3.jpg	Donostia-San Sebastián	Guipúzcoa	Museo
ARCHIVO MANUEL DE FALLA	No disponible	Granada	Granada	Museo
ARCHIVO MUSEO DON ÁLVARO DE BAZÁN	No disponible	Viso del Marqués	Ciudad Real	Museo
ARTIUM MUSEOA. MUSEO DE ARTE CONTEMPORÁNEO DEL PAÍS VASCO	No disponible	Vitoria-Gasteiz	Álava	Museo
ARXIU I MUSEU DE L'EDUCACIÓ DE LES ILLES BALEARS	descargarFichero.do?fileName=201/thumb_logo museu educació .tif color.JPG	Inca - Mallorca	Illes Balears	Museo
ALBAOLA LA FACTORÍA MARÍTIMA VASCA	No disponible	Pasaia	Guipúzcoa	Museo
ALMA MATER MUSEUM. MUSEO DIOCESANO	No disponible	Zaragoza	Zaragoza	Museo

Museos y Colecciones

Museos | Colecciones

Nombre	Imagen	Municipio	Provincia	Tipo
ALMAZARA DEL CONDE	No disponible	Sot de Chera	València/Valencia	Colección
ARCHIVO - MUSEO IGNACIO SÁNCHEZ MEJÍAS	No disponible	Manzanares	Ciudad Real	Colección
ART MAJOR DE LA SEDA	descargarFichero.do?fileName=1421/thumb_FAMA.jpg	Valencia	València/Valencia	Colección
ASOCIACION CULTURAL SANTISIMO CRISTO DE LA VERA CRUZ	No disponible	Consuegra	Toledo	Colección
ASOCIACIÓN DE ESTUDIOS MELILLENSES	No disponible	Melilla	Melilla	Colección
AULA DIDÁCTICA DEL CASTRO DE COAÑA	descargarFichero.do?fileName=1080/thumb_Coaña.jpg	Coaña	Asturias	Colección
ALMAZARA DEL CONDE	No disponible	Sot de Chera	València/Valencia	Colección
ARCHIVO - MUSEO IGNACIO SÁNCHEZ MEJÍAS	No disponible	Manzanares	Ciudad Real	Colección
ART MAJOR DE LA SEDA	descargarFichero.do?fileName=1421/thumb_FAMA.jpg	Valencia	València/Valencia	Colección
ASOCIACION CULTURAL SANTISIMO CRISTO DE LA VERA CRUZ	No disponible	Consuegra	Toledo	Colección
ASOCIACIÓN DE ESTUDIOS MELILLENSES	No disponible	Melilla	Melilla	Colección
AULA DIDÁCTICA DEL CASTRO DE COAÑA	descargarFichero.do?fileName=1080/thumb_Coaña.jpg	Coaña	Asturias	Colección
ALMAZARA DEL CONDE	No disponible	Sot de Chera	València/Valencia	Colección

Figura 3-9 museos_y_colecciones.html

3.4 EMF

El proceso comienza con la definición del metamodelo en el archivo Museos.ecore [7]. En este archivo, se ha establecido la estructura del modelo que describe la organización de museos en España según sus provincias. El metamodelo define dos entidades principales: Provincia y Museo. Una Provincia puede contener múltiples instancias de Museo, representando una relación de contención entre ambos.

La entidad Museo cuenta con varios atributos: nombre, tipo, imagen, muncipio y provincia. Este diseño permite organizar los museos según sus provincias, de manera que cada provincia contenga una lista de museos y colecciones asociadas. El modelo sigue una estructura clara y bien definida, permitiendo una representación jerárquica y navegable de los datos museísticos. Además, para que el XML pueda ser leído correctamente, se creará una entidad llama Museos, la cual tiene una lista de tipo "Museo". Además de modificar las propiedades de cada elemento para darle el tipo de dato correspondiente, en aquellas que son listas hay que modificar la propiedad "Upper Bound" y darle valor -1, y añadir que un "Museo" solo puede pertenecer a una "Provincia" pero una "Provincia" puede contener muchos "Museo".

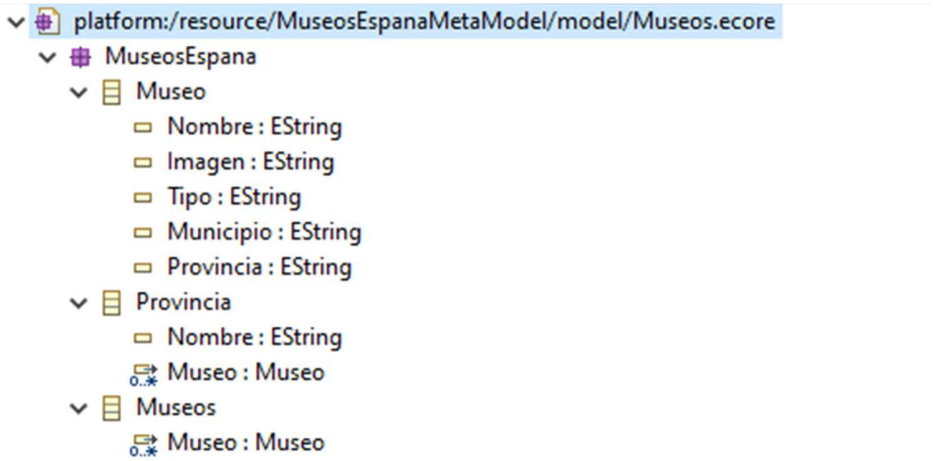


Figura 3-19 Museos.ecore

Es muy importante, que a MuseosEspana se le asignen correctamente las propiedades. El Name y Ns PRefix deben de llamarse "MuseosEspana" y Ns URI tiene que ser un http que sea único, para ellos se le asigna uno que no existe.

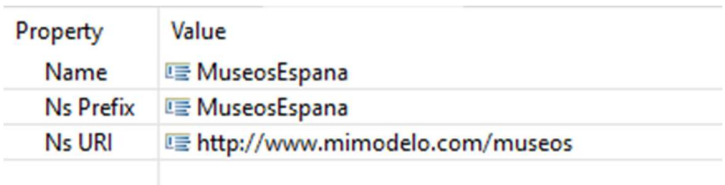


Figura 3-20 Propiedades MuseosEspana

El archivo Museos.genmodel, generado a partir del metamodelo, se encarga de producir el código Java correspondiente, incluyendo clases y métodos para manipular instancias del modelo, esto se genera ejecutando el archivo .genmodel en modo "Generate All" . Esto permite cargar, modificar y guardar estructuras completas basadas en la definición del metamodelo.

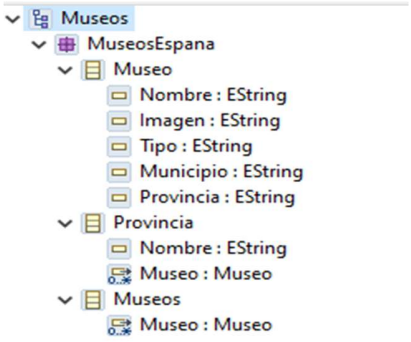


Figura 3-21 Museos.genmodel

En el archivo "GeneradorJSON.java" esta implementado el código para cargar el archivo "museos.xml" y generar el archivo JSON como solución.

Este código comienza creando "ResourceSet" una estructura en EMF que se encarga de gestionar y almacenar los recursos. A continuación, se configura el " ResourceSet" para cargar el archivo utilizando XMLResourceFactoryImpl, una clase que interpreta y lee archivos en formato XML. Una vez cargado el archivo, se accede al contenido mediante la lista "contents" que se obtiene del ResourceSet.

Para generar el archivo JSON, se ha añadido la librería Jackson. Primero, se crea un JsonFactory y un JsonGenerator. El JsonGenerator es el encargado de escribir en el archivo "museos.json" en la ruta especificada mediante la llamada a la función jsonGenerator.writeStartObject(). A continuación, se agrupan los museos por provincia. El código recorre todos los elementos, y para cada museo, obtiene la "provincia" a la que está provincia, y se inicia un array de los museos a los que pertenece. Además, para cada museo, se escriben sus atributos como el "nombre", "municipio", "imagen" y "provincia". Estos atributos son extraídos del modelo "museo" y se escriben en el "museos.json" usando JsonGenerator. Por último, hay que modificar el archivo "museos.xml" y añadir en <Museos> el Ns URI de nuestro metamodelo. <Museos

xmlns="http://www.mimodelo.com/museos">. En la carpeta “resources” tiene que estar ubicado el archivo “museos.xml” y se generará ahí la solución “museos.json”.

```
public class GeneradorJSON {
    public static void main(String[] args) {
        try {
            ResourceSet resourceSet = new ResourceSetImpl();
            resourceSet.getResourceFactoryRegistry().getExtensionToFactoryMap().put("xml", new XMLResourceFactoryImpl());
            resourceSet.getPackageRegistry().put(MuseosEspanaPackage.eNS_URI, MuseosEspanaPackage.eINSTANCE
            );
            // Cargar el archivo XML
            URI fileURI = URI.createFileURI("C:/Users/Paula/eclipse-workspace/MuseosEspanaMetaModel/resources/museos.xml");
            Resource resource = resourceSet.getResource(fileURI, true);
            resource.load(null);
            List<EObject> contents = resource.getContents();
            Museos museos = (Museos) contents.get(0);
            // Crear el JsonFactory para generar el archivo JSON
            JsonFactory jsonFactory = new JsonFactory();
            File outputFile = new File(System.getProperty("user.dir") + "/resources/museos.json");
            // Crear un JsonGenerator para escribir el archivo JSON
            JsonGenerator jsonGenerator = jsonFactory.createGenerator(outputFile, com.fasterxml.jackson.core.JsonEncoding.UTF8);
            // Iniciar la escritura del objeto JSON
            jsonGenerator.writeStartObject();
            // Mapa para almacenar los museos agrupados por provincias
            Map<String, EList> mapaProvincias = new HashMap();
            for (Museo museo: museos.getMuseo()) {
                    String provincia = museo.getProvincia();
                    // Si la provincia aún no está en el mapa, la agregamos
                    if (!mapaProvincias.containsKey(provincia)) {
                        EList museoList = new BasicEList();
                        museoList.add(museo);
                        mapaProvincias.put(provincia, museoList);
                    }
                    else {
                        mapaProvincias.get(provincia).add(museo);
                    }
```

```
            }
            // Escribir los museos agrupados por provincia en el JSON
            for(String provincia: mapaProvincias.keySet())  {
                // Escribir la provincia como clave en el JSON
                jsonGenerator.writeFieldName(provincia);
                // Comienza un array para esa provincia
                jsonGenerator.writeStartArray();
                EList<Museo> museosProvincia = mapaProvincias.get(provincia);
                // Escribir los museos dentro de la provincia
                for (Museo museo: museosProvincia) {
                    jsonGenerator.writeStartObject();
                    jsonGenerator.writeStringField("Nombre", museo.getNombre());
                    jsonGenerator.writeStringField("Municipio", museo.getMunicipio());
                    jsonGenerator.writeStringField("Imagen", museo.getImagen());
                    jsonGenerator.writeStringField("Provincia", museo.getProvincia());
                    jsonGenerator.writeEndObject();
                }
                jsonGenerator.writeEndArray();
            }

            jsonGenerator.writeEndObject();
            jsonGenerator.close();
            System.out.println(outputFile.getAbsolutePath());
        } catch (IOException e) {
            e.printStackTrace();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Figura 3-22 GeneradorJSON.java

```
{
    "\"Madrid\"": [
        {
            "Nombre": "\"AMBAJA (AMIGOS DEL MUSEO DE LA BATALLA DEL JARAMA)\"",
            "Municipio": "\"Morata de Tajuña\"",
            "Imagen": "\"No disponible\"",
            "Provincia": "\"Madrid\""
        },
        {
            "Nombre": "\"BASÍLICA DE LA SANTA CRUZ DEL VALLE DE LOS CAÍDOS\"",
            "Municipio": "\"San Lorenzo de El Escorial\"",
            "Imagen": "\"No disponible\"",
            "Provincia": "\"Madrid\""
        },
        {
            "Nombre": "\"CALCOGRAFÍA NACIONAL\"",
            "Municipio": "\"Madrid\"",
            "Imagen": "\"No disponible\"",
            "Provincia": "\"Madrid\""
        }
    ],
    "\"Girona\"": [
        {
            "Nombre": "\"CAN QUINTANA. CENTRE CULTURAL  DE LA MEDITERRÀNIA\"",
            "Municipio": "\"Torroella de Montgrí\"",
            "Imagen": "\"No disponible\"",
            "Provincia": "\"Girona\""
        }
```


Figura 3-23 museo.json

Capítulo 4 - Diferencias entre Xtext y Xtend contra EMF en el proyecto.

A continuación, se describen las diferencias que se han ido obteniendo durante el desarrollo del proyecto en los lenguajes Xtext y Xtend contra EMF a la hora de ponerlos en práctica. A través de ejemplos concretos, se explica cómo cada uno influyó en la implementación de las funcionalidades, y como sus características afectaron en el desarrollo, la eficiencia y la solución de los problemas.

4.1 Uso de EMF en el proyecto

4.1.1 Modelado estructurado con EMF

Una de las principales ventajas de este lenguaje en mi proyecto fue la capacidad de crear y gestionar modelos estructurados. El metamodelo "Museos.ecore" me permitió definir de manera clara y sencilla las entidades fundamentales, como "Museo", "Provincia" y "Museos", con sus respectivos atributos y relaciones, así como poder marcar la división por provincias que iba a haber en la solución. Pude crear el metamodelo de datos de los museos con los atributos como "Nombre", "Imagen", "Tipo", "Municipio" y "Provincia", que se ven de forma esquemática, y poder organizarlos según la "Provincia" a la que corresponden, además de crear "Museos" para que se pueda leer el XML correctamente.

La importancia de este lenguaje es que genera automáticamente las clases Java correspondientes basándose en mi metamodelo, creando cada clase con sus atributos, relaciones y métodos básicos. Además, no es necesario escribir manualmente en los métodos los accesos a los atributos ni las relaciones entre clases (por ejemplo, la relación que hay entre un "Museo" y una "Provincia").

```java
/**
 */
package MuseosEspana;

import org.eclipse.emf.common.util.EList;

import org.eclipse.emf.ecore.EObject;

/**

 */
public interface Provincia extends EObject {
    /**

     */
    String getNombre();

    /**

     */
    void setNombre(String value);

    /**

     */
    EList<Museo> getMuseo();

} // Provincia
```

Figura 4-1 Provincia.java

4.1.2 Lectura de archivos con EMF

Una de las funcionalidades que considero importante, que conseguí implementar con el lenguaje EMF mientras que con Xtext y Xtend no lo conseguí, es el poder leer el archivo "museos.xml" ubicado en local, y desde el GeneradorJson.java donde realice este lectura poder obtener los datos y almacenarlos en el archivo JSON. Si no hubiese realizado esta implementación, habría tenido que escribir todo el código para leer y de serializar el XML manualmente, lo que me podría haber supuesto más trabajo y errores.

4.2 Uso de Xtext y Xtend en el proyecto.

4.2.1 Creación de un lenguaje específico de dominio

Con Xtext pude crear un lenguaje específico de dominio para describir los museos de una forma más intuitiva y accesible. Diseñe una sintaxis propia a mi descripción de los museos "MyDsl.xtext", y aunque a la hora de realizar modificaciones, en esta

estructura me resultó más sencillo que en EMF, no pude conseguir declarar las divisiones que iba a tener mi solución en base a qué "TipoMuseo" era cada uno, ni dividirlo en "Provincias", como había hecho en el anterior lenguaje. Estas divisiones de "TipoMuseo" iban a estar en el código "MyDslGenerator.xtend", pero esto implica un trabajo más tedioso y cometer más errores a la hora de implementarlo comparado con EMF, donde como expliqué anteriormente en el archivo "Museos.ecore" se añadían estas divisiones.

4.2.2 Lectura de archivos con Xtend

Con Xtend, me encontré con dificultades para poder leer el archivo "museos.xml" desde mi escritorio y poder implementar dicha lectura en el código del archivo "MyDslGenerator.xtend", como sí pude hacer en EMF. Esto supone que tengas que ejecutar el código y después crear un archivo en blanco con el nombre de tu .xtext y copiar todo el contenido de tu fichero XML para que pueda construir la solución. Esto es mucho más costoso y menos eficiente.

4.3 Conclusión

Después de trabajar con ambos lenguajes en este proyecto, llegué a la conclusión de que el lenguaje EMF fue la más adecuada para gestionar los datos y manejar modelos estructurados. Por otra parte, Xtext y Xtend resultaron útiles para crear un lenguaje específico de dominio y transformar los datos a HTML, aunque no resultaron buena opción para gestionar de manera eficiente los datos y crear archivos JSON. Con EMF, como mencioné anteriormente, pude definir de manera clara la relación entre "Museo" y "Provincia" en el archivo .ecore, lo que me permitió estructurar los datos sin tener que escribir código adicional. Por el contrario con Xtend era más complicado, aunque he de destacar que el archivo "MyDsl.xtext" es más fácil de modificar.

Además, EMF me facilitó la lectura de mi archivo "museos.xml" utilizando el recurso "Resource", permitiéndome cargar y procesar los datos de forma automática. Mientras tanto, con "Xtext" y "Xtend", tuve que copiar manualmente los datos del archivo XML.

Mi conclusión es que el lenguaje EMF fue claramente la opción más eficiente y adecuada para gestionar los datos de manera estructurada y poder personalizar mi

solución, así como para interactuar con mi archivo XML. Xtext y Xtend, aunque han sido útiles para tareas como la creación de un DSL y la modificación visual de los datos, no fueron lo suficientemente buenos para manejar de forma óptima la gestión, clasificación y almacenamiento de los datos.

Capítulo 5 - Conclusiones y trabajo futuro

5.1 Conclusiones

Este proyecto ha demostrado la posibilidad de poder desarrollar diferentes formatos de soluciones relacionados con datos de los museos españoles, utilizando los lenguajes de modelado y automatización, siendo capaz de lograr automatizar procesos y dejando un espacio claro y sencillo para futuros desarrolladores. Así mismo, se ha demostrado la facilidad que obtendrán los dueños de los museos para desarrollar sus aplicaciones utilizando dichos lenguajes y la gran cantidad de datos que pueden procesar.

Los lenguajes Xtext, Xtend y EMF y las librerías de Python para realizar scraping y generar los XML, han resultado proporcionar una base sólida y escalable para la solución. Dichos lenguajes, permiten que los dueños de los museos sean capaces de modificar la estructura del proyecto de forma sencilla sin tener que ser expertos en programación, además de que este diseño permite el crecimiento gradual ya que se adapta a las necesidades cambiantes de los museos.

Es importante reconocer que aunque el conocimiento de programación ha de ser menor que en otros lenguajes, aun así requiere un mínimo de conocimiento técnico para poder modificar y personalizar el proyecto. Si bien se ha buscado minimizar esta dependencia a través de la automatización de los procesos y el modelado.

También hay que destacar, que a futuros diseñadores dichos lenguajes son muy útiles para poder construir soluciones más complejas de una forma más sencilla.

5.2 Trabajo futuro

En base a los resultados que he obtenido en el proyecto, se proponen las siguientes ideas para poder mejorar y ampliar dicho proyecto:

Primero, adaptar a otros formatos de entrada de datos más avanzados, ya que en este proyecto se utilizó únicamente XML, dichos formatos podría facilitar el intercambio de datos entre plataformas y conectar de diferentes maneras la

información. Esto proporcionaría que dicho proyecto se pudiese integrar con otras herramientas y que funcione de manera más eficiente con diferentes bases de datos.

También muy útil desarrollar una interfaz gráfica fácil de usar, para que cualquier persona, incluso sin conocimientos técnicos, pueda personalizar su página web y modificar sus colecciones de manera visual y práctica.

Otra propuesta, y en la cual más énfasis doy es en la de agregar funcionalidades que permitan crear recorridos virtuales en 3D o aplicaciones de realidad aumentada, unas soluciones que en la actualidad se están viendo muy populares y llamativas. Además, de usar herramientas de inteligencia artificial para ofrecer a los usuarios soluciones personalizadas según sus gustos, creando así las visitas virtuales más atractivas. Esta propuesta, está muy en tendencia en la actualidad.

Otra mejora importante, sería poder optimizar el lenguaje de Python donde se realizó Scraping para que esta sea más rápida y pueda manejar grandes cantidades de datos sin problemas.

Para terminar, también sería valioso poder fomentar esta forma de desarrollo a una comunidad de programadores, para que puedan compartir sus ideas, experiencias y conocimientos. Este hecho mejoraría y aceleraría su desarrollo y garantizaría que sea útil y sostenible a largo plazo.

Construyendo estas mejores, se logaría una herramienta clave para digitalizar y disfrutar del patrimonio cultural, ayudando a que futuras generaciones se interesen más y además sea accesible para todos.

Introduction

As technologies advance today, museums must adapt to these advances in order to generate revenue and achieve a high number of visits, which is a great challenge for many of them, especially for those museums that have limited resources and little knowledge of programming. In addition, there is currently a large number of users who prefer such innovative technologies as augmented reality or virtual tours to increase their

cultural information. This project allows museums with few economic and technical resources to develop their applications in a simpler way.

The source code of this project is available at the following link:

https://drive.google.com/drive/folders/1P8sW2anUBObLy1hoY_aZGFzZlhd5MbjX?dmr=1&ec=wgc-drive-%5Bmodule%5D-goto

Motivation

Nowadays [1,2], most museums are facing the challenge of adapting their collections to technological formats, since technology is booming and more and more innovative technologies are being added. This means that museums have to integrate with these technologies in order to reach a wider and more diverse audience. Due to the different ways in which people perceive and enjoy these collections, it is necessary to use more flexible methods adapted to their needs.

It is a fact, that future generations enjoy more cultural collections which provide 3D technologies or augmented reality, likewise, there is a great part of people who need to know if museums are adapted to their needs in order to enjoy them without any complication. This causes the need to adapt to these updates, which means that many museums may not have the economic and technical resources to adapt to these needs.

For example, institutions such as the Louvre Museum have introduced virtual tours and mobile applications to assist and enhance the experience of visitors with reduced mobility or sensory diversity. Such an example shows how valuable it is to have tools that can adapt to different situations and adjust to what is needed, but it is a big problem for those museums which do not have such resources as important museums.

This project arises with the purpose of making it easier for museums to develop customized websites by reducing the need for advanced technical knowledge. It has been done through model-driven development techniques and automatic code generation. Museums will be able to create HTML web pages or JSON files by entering only the data of their collections. This simplifies the process and significantly reduces the technical and economic effort required.

In addition, the automation of the process allows for an incremental approach, as museums can start with basic solutions and add more advanced functionalities, such as 3D models, virtual tours or augmented reality, depending on what their resources allow.

This project has been developed using the Xtext, Xtend and EMF tools in Eclipse, Python libraries such as BeautifulSoup and ElementTree to process the data and generate the XML files and build the respective databases. The combination of these technologies offers a complete and automatic solution, allowing museums to create their websites from structured data, without the need for advanced programming skills once the necessary infrastructure is in place.

In this way, the project helps museums to make their content accessible and enjoyable for all users, promoting inclusion and technological innovation in the museum sector..

Goals

This project aims to create an automated system that allows museums to generate HTML and JSON files, in order to create and customize their web pages, without the need for them to have high technical knowledge. The main objective is to facilitate and help the publication of their works, and attract more users to enjoy and interact with them.

The system automatically creates files with added data such as descriptions, images, province, key data for them, allowing users to know if a museum is suitable or interesting for them before visiting it. In addition, it is planned to include virtual tours, 3D models and augmented reality in the future.

The main objectives are:

- Data management: Organize museum information, such as collections, subjects and highlights, in order to be able to work with correctly structured data and the necessary information.

- Accessibility to users: Create clear web pages, easy to understand and adapted to different users.

- Ease of use for museums: Allow museums to publish or add information about themselves, reducing their need for programming knowledge.

- Interactivity and design: Display information in an attractive and user-friendly way for different users.

- Future expansion: Prepare the system to incorporate new functions such as mobile applications and 3D models.

This project seeks to help people's access to cultural information and improve their experiences in them, while helping museums to reach more people and facilitate the development of their applications.

Work plan

1. Preliminary research (1-2 months): The objective of this first phase was to investigate about the different Eclipse Modeling Tools languages, their operation, complexity and the different results provided by each one. In addition, the search of the different web pages about museums, the data provided by each of them, such as the complexity of their code.

2. Extracting the information from the web pages (3 – 4 weeks): This second phase consisted in learning Web Scraping in Python, and use it to extract the information from the two selected web pages and store them in an XML.

3. Development of Xtext, Xtend and EMF languages (2 – 3 months): This last phase consisted of learning and developing the Xtext, Xtend and EMF languages, which return HTML web pages and JSON files as a solution.

Conclusions and future work

Conclusions

This project has demonstrated the possibility of being able to develop different formats of solutions related to data from Spanish museums, using modeling and automation languages, being able to automate processes and leaving a clear and simple space for future developers. Likewise, it has been demonstrated how easy it will be for museum owners to develop their applications using these languages and the large amount of data they can process.

The Xtext, Xtend and EMF languages and the Python libraries for scraping and XML generation have proven to provide a solid and scalable foundation for the solution. These languages allow museum owners to be able to easily modify the project structure without having to be programming experts, and the design allows for gradual growth as it adapts to the changing needs of the museums.

It is important to recognize that although the programming knowledge has to be less than in other languages, it still requires a minimum of technical knowledge to be able to modify and customize the project. Although we have sought to minimize this dependency through process automation and modeling.

It should also be noted that these languages are very useful for future designers to be able to build more complex solutions in a simpler way.

Future Work

Based on the results I have obtained in the project, the following ideas are proposed to improve and expand the project:

First, adapt to other more advanced data entry formats, since only XML was used in this project, such formats could facilitate the exchange of data between platforms and

connect in different ways the information. This would provide that such a project could be integrated with other tools and work more efficiently with different databases.

It would also be very useful to develop a user-friendly graphical interface, so that anyone, even without technical knowledge, could customize their web page and modify their collections in a visual and practical way.

Another proposal, and the one I emphasize the most, is to add functionalities that allow the creation of 3D virtual tours or augmented reality applications, solutions that are currently becoming very popular and eye-catching. In addition, using artificial intelligence tools to offer users customized solutions according to their tastes, thus creating the most attractive virtual tours. This approach is very much on trend at present.

Another important improvement would be to be able to optimize the Python language where Scraping was performed so that it is faster and can handle large amounts of data without problems.

Finally, it would also be valuable to be able to encourage this form of development to a community of programmers, so that they can share their ideas, experiences and knowledge. This would enhance and accelerate its development and ensure that it is useful and sustainable in the long term.

By building these best practices, a key tool for digitizing and enjoying cultural heritage would be achieved, helping future generations to become more interested in it and making it accessible to all.

BIBLIOGRAFÍA

[1] J. Cueva, Model-driven engineering, http://di002.edv.uniovi.es/~cueva/asignaturas/masters/2008/MDE_udistrital.pdf.

[2] LifeArTech, MDE: Model Driven Engineering (Ingeniería dirigida por modelos), https://lifeartech.wordpress.com/2017/08/28/mde-model-driven-engineering-ingenieria-dirigida-por-modelos/.

[3] Real Python, Beautiful Soup: Web Scraper en Python, https://realpython.com/beautiful-soup-web-scraper-python/.

[4] Universidad de Cádiz, Desarrollo de editores textuales con Xtext, https://ocw.uca.es/pluginfile.php/2499/mod_resource/content/0/P6%20-%20Desarrollo%20de%20editores%20textuales%20con%20Xtext.pdf.

[5] Eclipse Foundation, Xtext Documentation: Runtime Concepts, https://eclipse.dev/Xtext/documentation/303_runtime_concepts.html

[6] Uqbar, Xtend: Creación de Proyecto, https://wiki.uqbar.org/wiki/articles/xtend-creacion-proyecto.html

[7] Eclipsesource, EMF Tutorial, https://eclipsesource.com/blogs/tutorials/emf-tutorial/.

[8] EVE Museografía, "Museos digitales: Retos y oportunidades,", https://evemuseografia.com/2020/09/25/museos-digitales-retos-y-oportunidades/.

[9] EVE Museografía, "Retos de los museos digitales,", https://evemuseografia.com/2020/09/18/retos-de-los-museos-digitales/.

10 Europeana, "Themes,", https://www.europeana.eu/es/themes.

11 Europeana, "Features,", https://www.europeana.eu/es/features.

12 Ministerio de Cultura y Deporte, "Directorio de Museos,", https://directoriomuseos.mcu.es/dirmuseos/realizarBusquedaGeneral.do.