5. 1. TRANSFORMACIÓN DE LOS DATOS Y PERSONALIZACIÓN DE ESQUEMAS

5. 1. 1. Transformando TEI con OxGarage

OxGarage es una aplicación web de código abierto (http://oxgarage.oucs.ox.ac.uk:8080/ege-webclient) que proporciona transformaciones de documentos, que ofrece Interfaz Web y REST, conversiones encadenadas XSLT y usa TEI XML como formato de pivote.

En el ejemplo 5. 1. trabajaremos en la conversión de archivos por medio de OxGarage.

5. 1. 2. Una rápida introducción a XSLT

XPath es un lenguaje para recorrer, hacer búsquedas y expresar caminos a través de los árboles de los documentos XML. Los documentos XML son tratados como árboles de nodos. El elemento más alto en el árbol se llama elemento raíz.

```
<?xml version="1.0" encoding="UTF-8"?>
<prosaLatina>
        <libro>
        <título lang="la">Ab urbe condita</título>
        <autor>Tito Livio</autor>
        <fecha>27 a. C.</fecha>
        </libro>
        <libro>
        <título lang="la">De uita XII Caesarum</título>
        <autor>Gayo Suetonio Tranquilo</autor>
        <fecha>121 d. C.</fecha>
        </libro>
</prosaLatina>
```

En el ejemplo anterior <prosaLatina> es el elemento raíz, <autor> es elemento nodo y lang: "la" es atributo nodo. Valores atómicos son nodos que no tienen hijos o padres (Gayo Suetonio Tranquilo, o "la"). Las relaciones entre los nodos son las siguientes: cada elemento y atributo tiene un padre (en nuestro ejemplo el elemento <libro> es padre de <título>, <autor> y <año>, o lo que es lo mismo, estos últimos son hijos del elemento <libro>). Los nodos que tienen el mismo padre se llaman hermanos (en nuestro ejemplo, son hermanos <título>, <autor> y fecha>. Se llama ancestros al padre de un nodo, o al padre de su padre (en nuestro ejemplo, los ancestros del elemento <título> son <libro> y <prosaLatina>). Descendientes son los hijos de un nodo, o los hijos de los hijos (en nuestro ejemplo, los descendientes del elemento <prosaLatina> son <libro>, <título>, <autor> y <fecha>).

XPath usa expresiones path para seleccionar nodos o conjuntos de nodos en un documento XML. El nodo se selecciona siguiendo un camino o unos pasos. En la tabla que sigue presentamos algunas expresiones path y el resultado de la búsqueda.

prosaLatina	Selecciona todos los nodos con el nombre prosaLatina
/prosaLatina	Selecciona el elemento raíz prosaLatina. El símbolo / representa un camino directo a un elemento.
prosaLatina/libro	Selecciona todos los elementos libro que son hijos de prosaLatina
//libro	Selecciona todos los elementos libro, independientemente de dónde

	estén en el documento.
prosaLatina//libro	Selecciona todos los elementos libro que son descendientes de prosaLatina, no importa dónde estén por debajo del elemento prosaLatina.
//@lang	Selecciona todos los atributos que se llaman lang

Los predicados se usan para encontrar un nodo específico o un nodo que contiene un valor específico. Los predicados son siempre encerrados en paréntesis cuadrados. En la tabla que sigue se muestran algunas expresiones path con predicados y el resultado de las expresiones.

/prosaLatina/libro[1]	Selecciona el primer elemento libro que es hijo del elemento prosaLatina
/prosaLatina/libro[last()]	Selecciona el último elemento libro que es hijo del elemento prosaLatina
/prosaLatina/libro[last()-1]	Selecciona el penúltimo elemento libro que es hijo del elemento prosaLatina
/prosaLatina/libro[last()<3]	Selecciona los dos primeros elementos libros que son hijos del elemento prosaLatina
//título[@lang]	Selecciona todos los elementos título que tienen un atributo llamado lang (lengua).
//title[@lang='la']	Selecciona todos los elementos título que tienen un atributo lengua con el valor "la" (latín)

XSLT (Extensible Stylesheet Language Tranformation) es una lengua de programación para transformar los documentos XML. El lenguaje XSLT está expresado en XML, usa espacios de nombre para distinguir los resultados de las instrucciones, es puramente funcional y se usa para leer y escribir documentos XML. Es empleado generalmente para producir HTML, pero puede usarse también para generar otros formatos como pdf, docx, etc.

¿Qué es una transformación?

Desde un documento XML tal como este:

```
<div type="receta" n="34">
  <head>Pasta para principiantes</head>
  <list>
    <item>Pasta</item>
    <item>Queso rallado</item>
  </list>
  <p>Cocina la pasta y mézclala con el queso</p>
</div>
```

Se puede llegar a un documento HTML como este:

```
<html>
<h1>34: Pasta para principiantes</h1>
<p>Ingredientes: Pasta, queso rallado</p>
<p>Cocina la pasta y mézclala con el queso</p>
</html>
```

Así se expresaría esto en XSLT:

```
<xsl:stylesheet xpath-default-namespace="http://www.tei-c.org/ns/1.0"
  version="2.0">
  <xsl:template match="div">
    <html>
      <h1>
        <xsl:value-of select="@n"/>:
        <xsl:value-of select="head"/>
      </h1>
      <p>Ingredients:
        <xsl:apply-templates select="list/item"/>
      </p>
      <p>
        <xsl:value-of select="p"/>
      </p>
    </html>
  </xsl:template>
</xsl:stylesheet>
```

El consorcio TEI posee y maneja una familia de hojas de estilo XSLT que operan en documentos TEI XML. Pueden usarse para implementar un procesador ODD, generando esquemas y documentación de fuentes TEI, para convertir documentos XML a formatos legibles por el hombre como HTML, ePub, LaTeX, XSL FO, para convertir entre TEI XML y Microsoft Word, y entre TEI y Open Office. Las hojas de estilo de TEI están disponibles en Sourceforge (https://sourceforge.net/projects/tei/files/Stylesheets/), en oXygen, como Debian packages (para Linux); en OxGarage y en Github (http://www.github.com/TEIC/Stylesheets).

Para hacer una transformación desde oXygen hay que buscar y clicar el botón del menú. La primera vez que se haga, se nos preguntará que escenario de transformación vamos a elegir:

Veamos la estructura de un documento XSL:

```
<xsl:stylesheet xpath-default-namespace="http://www.tei-c.org/ns/1.0"
  version="2.0">
  <xsl:template match="div">
    <!-- .... do something with div elements....-->
  </xsl:template>
  <xsl:template match="p">
    <!-- .... do something with p elements....-->
  </xsl:template>
</xsl:stylesheet>
```

"div" y "p" son expresiones XPath, que especifican qué parte del documento es afectada por la plantilla. Cualquier elemento que no empiece con xsl: en el cuerpo de la plantilla queda fuera de la transformación final.

Nuestras hojas de estilo comenzarán con dos importantes atributos en el elemento <stylesheet>:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet xpath-default-namespace="http://www.tei-c.org/ns/1.0" version="2.0">
</xsl:stylesheet>
```

Esto significa que en nuestras expresiones XPath, cualquier nombre de elemento sin espacio de nombre se asume que es del espacio de nombre de TEI. 2. Queremos usar la versión 2.0 de la especificación XSLT, lo que significa que debemos usar el procesador Saxon para nuestro trabajo.

XSLT es un lenguaje declarativo. Por ello, las hojas de estilo XSLT no se escriben como una secuencia de instrucciones, sino como una colección de plantillas (template rules). Cada plantilla establece cómo se transforma un determinado elemento (definido mediante expresiones XPath). La transformación del documento se realiza de la siguiente manera:

- El procesador analiza el documento y construye el árbol del documento.
- El procesador va recorriendo todos los nodos desde el nodo raíz, aplicando a cada nodo una plantilla, sustituyendo el nodo por el resultado.
- Cuando el procesador ha recorrido todos los nodos, se ha terminado la transformación.

Tomemos este fichero XML (un catálogo de CDs) como punto de partida de las explicaciones subsiguientes:

```
<?xml version="1.0" encoding="UTF-8"?>
<catálogo>
  <cd>
    <título>Empire Burlesque</título>
    <artista>Bob Dylan</artista>
    <país>USA</país>
    <compañía>Columbia</compañía>
    <prrecio>10.90</precio>
    <año>1985</año>
  </cd>
   <cd>
   <título>Hide your heart</título>
   <artista>Bonnie Tyler</artista>
   <país>UK</país>
   <compañía>CBS Records</compañía>
   <precio>9.90</precio>
   <año>1988</año>
  </cd>
```

```
.
.
</catálogo>
```

El elemento <xsl:template> se usa para construir plantillas. El atributo match es usado para asociar una plantilla con un elemento XML. El atributo match se puede usar también para definir una platilla para el documento XML. El valor del atributo match es una expresión XPath (por ejemplo, match= "/" comprende todo el documento). Ejemplo:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

  <xsl:template match="/">
    <html>
      <body>
        <h2>Mi colección de CDs</h2>
        <table border="1">
          <tr bgcolor="#9acd32">
            <th>Título</th>
            <th>Artista</th>
          </tr>
          <tr>
            <td>.</td>
            <td>.</td>
          </tr>
        </table>
      </body>
    </html>
  </xsl:template>

</xsl:stylesheet>
```

Dado que una XSL stylesheet es un documento XML, siempre comienza con una declaración XML (<?xml version="1.0" encoding="UTF-8"?>). El siguiente elemento, (<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">) define que este documento es una hoja de estilo XSLT (junto con el número de versión y el atributo de espacio de nombre XSLT). El elemento <xsl:template match="/"> define una plantilla. El atributo match= "/" asocia la plantilla con la raíz del documento fuente XML. El contenido dentro del elemento <xsl:template match="/"> define algo de HTML para el resultado final. Las últimas dos líneas definen el final de la plantilla y el final de la hoja de estilo.

El elemento <xsl:value-of> se puede usar para extraer el valor de un documento XML y añadirlo al resultado de la transformación.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

  <xsl:template match="/">
    <html>
      <body>
        <h2>Mi colección de CDs</h2>
        <table border="1">
```

```
        <tr bgcolor="#9acd32">
          <th>Title</th>
          <th>Artist</th>
        </tr>
        <tr>
          <td><xsl:value-of select="catálogo/cd/título"/></td>
          <td><xsl:value-of select="catálogo/cd/artista"/></td>
        </tr>
      </table>
    </body>
  </html>
</xsl:template>

</xsl:stylesheet>
```

El atributo select, en el ejemplo anterior, contiene una expresión XPath, que funciona como un navegador del fichero. Una barra / selecciona subdirectorios. El resultado de esta hoja de estilo sería que se muestra solo una línea de datos:

Mi colección de CDs

Título	Artista
Empire Burlesque	Bob Dylan

Para mostrar todos los registros se debe usar la instrucción <xsl:for-each>, que permite repetir la instrucción un número de veces determinado:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

  <xsl:template match="/">
    <html>
      <body>
        <h2>My CD Collection</h2>
        <table border="1">
          <tr bgcolor="#9acd32">
            <th>Título</th>
            <th>Artista</th>
          </tr>
          <xsl:for-each select="catálogo/cd">
            <tr>
              <td><xsl:value-of select="título"/></td>
              <td><xsl:value-of select="artista"/></td>
            </tr>
          </xsl:for-each>
        </table>
      </body>
    </html>
  </xsl:template>

</xsl:stylesheet>
```

Resultado:

Mi colección de CDs

Título	Artista
Empire Burlesque	Bob Dylan
Hide your heart	Bonnie Tyler
Greatest Hits	Dolly Parton
Still got the blues	Gary Moore
Eros	Eros Ramazzotti Etc.

También podemos filtrar la salida del documento XML añadiendo un criterio para seleccionar el atributo en el elemento for-each: por ejemplo, <xsl:for-each select="catálogo/cd[artista='Bob Dylan']">. Los operadores de filtro son:

= (igual)
!= (no igual)
< (menos que)
> (más que)

El elemento <xsl:sort> se usa para ordenar el resultado. El atributo "select" indica qué elemento XML categorizar:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

  <xsl:template match="/">
    <html>
      <body>
        <h2>Mi colección de CDs</h2>
        <table border="1">
          <tr bgcolor="#9acd32">
            <th>Título</th>
            <th>Artista</th>
          </tr>
          <xsl:for-each select="catalog/cd">
            <xsl:sort select="artist"/>
            <tr>
              <td><xsl:value-of select="título"/></td>
              <td><xsl:value-of select="artista"/></td>
            </tr>
          </xsl:for-each>
        </table>
      </body>
    </html>
  </xsl:template>

</xsl:stylesheet>
```

En este caso la lista se ordena por autor y orden alfabético.

El elemento <xsl:if> se usa para poner un test condicional al contenido del documento XML. El elemento <xsl:if> se coloca dentro del elemento <xsl:for-each>.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

  <xsl:template match="/">
    <html>
      <body>
        <h2>Mi colección de CDs</h2>
        <table border="1">
          <tr bgcolor="#9acd32">
            <th>Título</th>
            <th>Artista</th>
            <th>Precio</th>
          </tr>
          <xsl:for-each select="catálogo/cd">
            <xsl:if test="price &gt; 10">
              <tr>
                <td><xsl:value-of select="título"/></td>
                <td><xsl:value-of select="artista"/></td>
                <td><xsl:value-of select="precio"/></td>
              </tr>
            </xsl:if>
          </xsl:for-each>
        </table>
      </body>
    </html>
  </xsl:template>

</xsl:stylesheet>
```

El resultado sería una tabla que muestre todos los títulos, autores y precios de los CDs cuyo precio supere los 10 euros.

El elemento <xsl:choose> se usa en conjunción con <xsl:when> y <xsl:otherwise> para expresar múltiples tests condicionales.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

  <xsl:template match="/">
    <html>
      <body>
        <h2>Mi colección de CDs</h2>
        <table border="1">
          <tr bgcolor="#9acd32">
            <th>Título</th>
            <th>Artista</th>
          </tr>
          <xsl:for-each select="catálogo/cd">
            <tr>
              <td><xsl:value-of select="título"/></td>
              <xsl:choose>
```

```
            <xsl:when test="price &gt; 10">
              <td bgcolor="#ff00ff">
                <xsl:value-of select="artista"/></td>
            </xsl:when>
            <xsl:otherwise>
              <td><xsl:value-of select="artista"/></td>
            </xsl:otherwise>
          </xsl:choose>
        </tr>
      </xsl:for-each>
    </table>
  </body>
 </html>
</xsl:template>

</xsl:stylesheet>
```

El código anterior añade color rojo a la celda de los artistas CUANDO el precio de su Cd supere los 10 euros.

El elemento <xsl:apply-templates> aplica una plantilla al elemento actual o a los nodos hijos del elemento actual. Si añadimos un atributo select al elemento <xsl:apply-templates>, procesará solo el elemento hijo que coincida con el valor del atributo. Podemos usar el atributo select para especificar el orden en que son procesados los nodos hijos. Ejemplo:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
 xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

 <xsl:template match="/">
  <html>
   <body>
    <h2>Mi colección de CDs</h2>
    <xsl:apply-templates/>
   </body>
  </html>
 </xsl:template>

 <xsl:template match="cd">
  <p>
   <xsl:apply-templates select="título"/>
   <xsl:apply-templates select="artista"/>
  </p>
 </xsl:template>

 <xsl:template match="título">
  Título: <span style="color:#ff0000">
   <xsl:value-of select="."/></span>
  <br />
 </xsl:template>

 <xsl:template match="artista">
  Artista: <span style="color:#00ff00">
   <xsl:value-of select="."/></span>
  <br />
 </xsl:template>
```

```
</xsl:stylesheet>
```

El resultado sería el siguiente:

Mi colección de CDs

Título: Empire Burlesque
Artista: Bob Dylan

Título: Hide your heart
Artista: Bonnie Tyler

Título: Greatest Hits
Artista: Dolly Parton

5. 1. 3. Personalizar TEI con Roma

Para trabajar con un documento XML y aplicar un tipo concreto de marcado debemos tener un modelo de trabajo, una especie de guía que nos indique de qué modo debemos proceder para codificar este texto, de manera que si interviene más de una persona, todas lleven a cabo un marcado homogéneo.

Para ello, como se avanzó en el tema 3, necesitamos asociar nuestro documento XML con un esquema que sirve justamente para establecer el tipo de marcado que podemos aplicar a un documento XML. Los esquemas son ficheros autónomos que se asocian al fichero XML-TEI en su prólogo de la siguiente manera:

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-model href="http://www.tei-c.org/release/xml/tei/custom/schema/relaxng/tei_lite.rng"
schematypens="http://relaxng.org/ns/structure/1.0"?
```

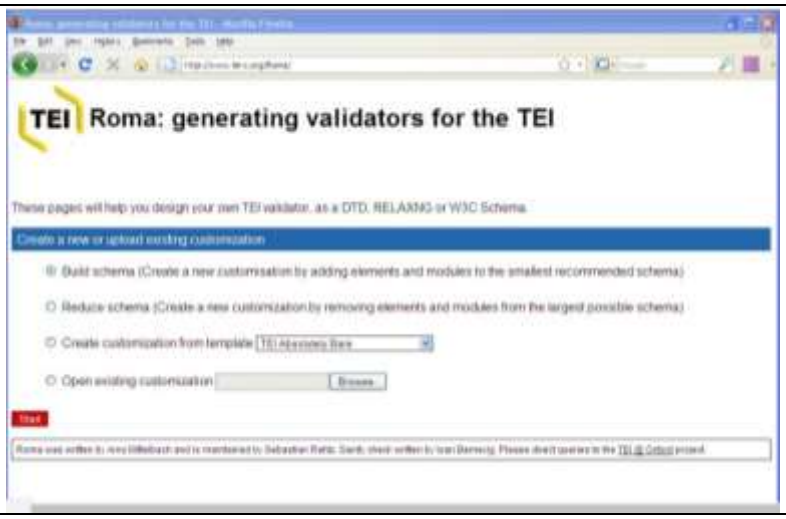
En general, los proyectos de edición digital no necesitan los 21 módulos ni todos los 505 elementos de las Guías directrices, de manera que es conveniente restringir y concretar lo más posible el esquema. Para crear nuestro esquema, la condición indispensable es que analicemos el texto y que definamos qué elementos aparecen en el mismo, tanto cuestiones estructurales como semánticas. Una vez hemos elaborado la lista de fenómenos que nos gustaría codificar, deberemos generar nuestro esquema Relax NG propio. Es fundamental al final de todo el proceso guardar el resultado de nuestra personalización en un documento ODD ("one document does it all"): se trata de un fichero XML que recoge todas las características del esquema personalizado y permite rehacerlo cuantas veces lo deseemos, conservando las modificaciones hechas.

La aplicación web Roma (http://www.tei-c.org/Roma/) proporciona una forma de especificar lo que cada uno quiere en un documento ODD sin tener que escribir el código XML. Roma no cubre todo lo que es posible en el lenguaje ODD, pero ofrece lo necesario para la mayoría de las ideas y para la personalización de TEI en particular. Se puede comenzar una sesión en Roma eligiendo un set mínimo de especificaciones ODD o cargando fichero ODD XML ya existente. Ello nos permite

personalizar nuestro fichero ODD, salvarlo y volver a cargarlo para nuevos cambios en una próxima sesión.

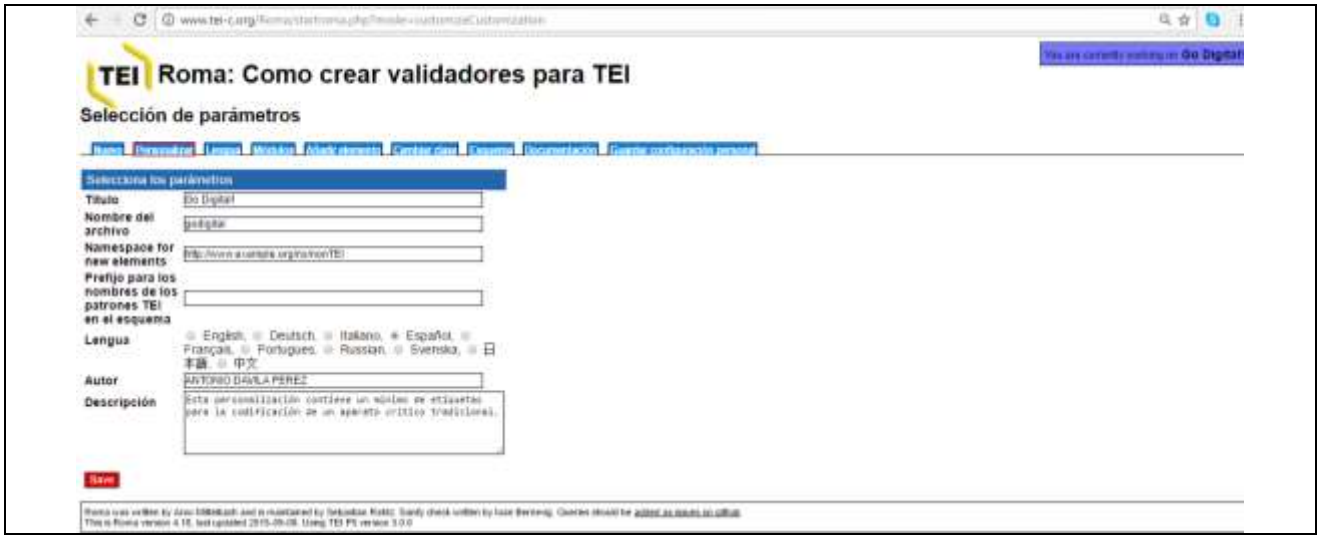
5. 1. 3. 1. Hacer un esquema con Roma

Lo primero que hay que hacer es abrir el navegador y visitar el sitio http://www.tei-c.org/Roma/. La pantalla inicial permite al usuario elegir de un número de puntos de partida: empezar desde un número mínimo de módulos TEI, a lo que podemos añadir nuevos elementos o módulos y quitar elementos existentes; comenzar desde el esquema TEI All y eliminar lo que no se quiera; comenzar desde un número de plantillas predefinidas; comenzar desde una personalización TEI ya existentes como la popular "TEI Lite"; y empezar desde una personalización que uno mismo u otra persona ha hecho ya.



En el ejercicio 5. 2. pedimos que se comience a partir de la primera opción (construir un esquema a partir del esquema mínimo). También podríamos empezar, como haremos en este tutorial, a partir de TEI Bare, y construir nuestra propia personalización añadiendo lo que necesitemos. Así pues, elegimos "Create customization from template" y presionamos el botón de "start".

En la siguiente pantalla hay que rellenar los siguientes parámetros: *Title* (cambiar al título de nuestro esquema); *Filename* (el nombre de este fichero XML no debe contener espacios); *Namespace* (dejar como está); *Language* (se puede elegir también español); *Author name* (teclea tu propio nombre); *Description* (describe brevemente el propósito de tu esquema). Haz clic en el botón "Save" al pie de la página. No olvides salvar frecuentemente.

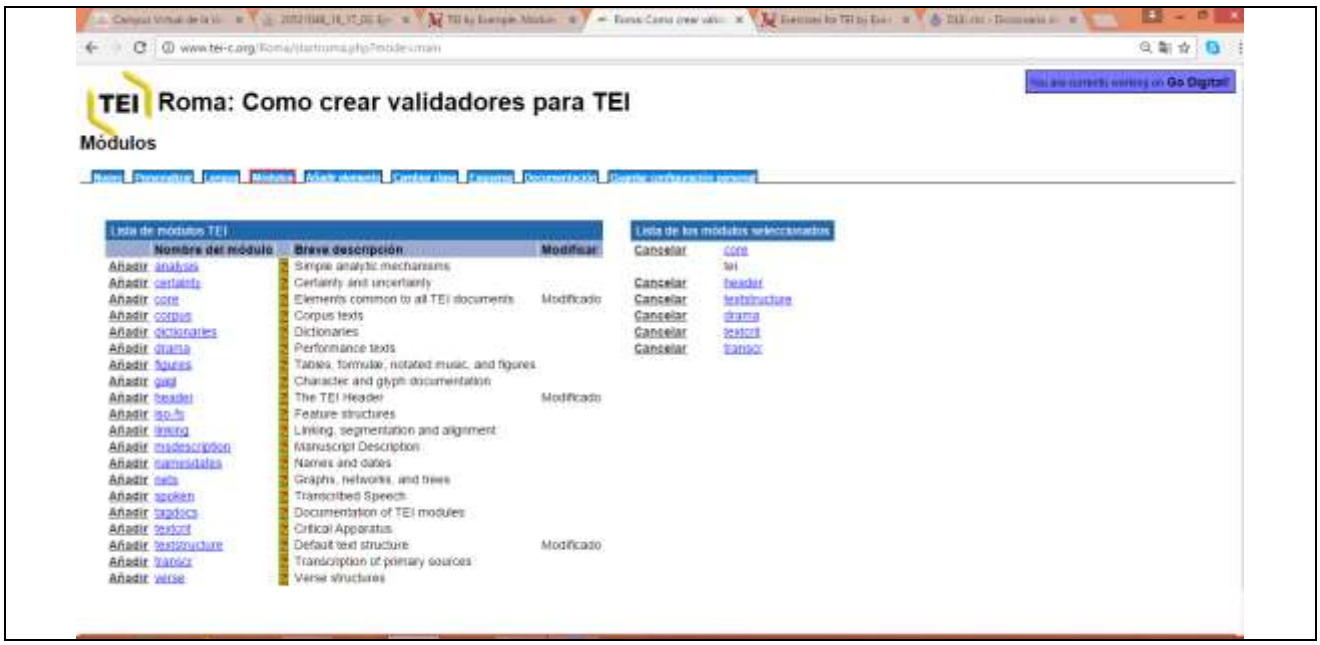


En el menú de la parte superior de la pantalla veras los siguientes botones: *New* (te lleva de nuevo a la pantalla de inicio para comenzar una nueva personalización); *Customize* (es donde nos encontramos ahora); *Language* (te permite elegir traducciones a distintas lenguas para el esquema y su documentación); *Modules* (te permite elegir las partes de TEI que necesitas); Add elements (te permite añadir tus propios elementos); *Change classes* (te permite cambiar y añadir atributos); *Schema* (te permite elegir qué tipo de esquema quieres generar); *Documentation* (te permite elegir qué tipo de formato de salida quieres para la documentación de tu esquema; *Save Customization* (te permite salvar tu personalización como un fichero ODD).

5. 1. 3. 2. Seleccionar los módulos

Un módulo es un grupo de elementos TEI. Cada elemento TEI está declarado en un módulo particular. Algunos módulos son muy generales y tienen muchos componentes; otros son más especializados. Por ejemplo, si vamos a codificar un diccionario tendríamos que elegir el módulo "Dictionaries", pero los elementos de ese módulo son de menor interés para otros tipos de documentos.

Haz click en el botón "Modules" de la barra de herramientas y mira el listado de módulos. La lista de módulos seleccionados por defecto se muestra a la derecha y contiene solo tres módulos (core, header y textstructure). Para el ejemplo 5. 2. nosotros hemos añadido los módulos "drama", "textcrit" y "transcr".



5. 1. 3. 3. Incluir y excluir elementos y atributos

Al incluir un módulo se incluye por defecto todos los elementos definidos por ese módulo, lo cual puede que no nos interese. Si queremos suprimir del módulo "trancr" los elementos <facsímile>, <surface> y <zone>, como es el caso del ejemplo 5. 2, habría que hacer clic en la palabra "transcr" de la lista de módulos seleccionados (a la derecha) y se nos mostraría una tabla con todos los elementos de ese módulo con información sobre: el nombre del elemento; una indicación sobre su inclusión o exclusión en el esquema; el nombre de este elemento en el esquema actual (Roma te permite darle un nuevo nombre a los elementos, por ejemplo, si estás trabajando en otra lengua que no sea el inglés); un signo de interrogación que enlaza con la información completa sobre ese elemento; una breve descripción del elemento; y un enlace que te permite cambiar los atributos del elemento.

Esta interfaz te permite explorar en detalle todos los elementos y, al mismo tiempo, controlar tu propia selección. Usa los botones de "incluir" y "excluir" para incluir o incluir por defecto todos los elementos definidos en el módulo. Si solo quieres trabajar con algunos elementos, haz clic en el botón "incluir" o "excluir" que se encuentra junto a los elementos en cuestión.

Dentro de cada elemento, el usuario puede operar de la misma manera con sus atributos (incluir, excluir y cambiar de nombre).

No olvides salvar.

5. 1. 3. 4. Crear un esquema

Haz clic en "Schema". Se puede elegir entre varias lenguas de esquema. Se recomienda generar el nuevo esquema en RELAXNG, ya sea en sintaxis compacta o XML. Haz clic en el botón rojo "Generar" y salva el fichero del esquema en tu carpeta de trabajo.

Recuerda que el esquema que has generado se puede utilizar como marco de trabajo en cualquier editor XML, y que por su mismo diseño permitirá que aparezcan unos elementos y otros no.

5. 1. 3. 5. Generar documentación

Cada proyecto necesita una documentación interna más discursiva y explicativa que un esquema RELAXNG. Roma se puede usar para generar esa documentación automáticamente. Regresa a Roma y pincha en el menú "Documentación". Elige a continuación el formato de salida de esa documentación (HTML, Word, etc.). El fichero generado por Roma, que podemos pasar a PDF si lo deseamos, contiene, a modo de guía o manual de trabajo, una explicación detallada de todos los elementos elegidos en nuestro esquema. Se puede modificar esta documentación, incluyendo, por ejemplo, ejemplos tomados de tu propio material de trabajo o eliminando comentarios irrelevantes. Para ello debes acceder al código fuente ODD que se puede obtener también en Roma.

5. 1. 3. 6. Salvar una configuración

Un fichero de personalización (un ODD) es un documento TEI como otro cualquiera. Se puede salvar y reeditar usando cualquier editor XML. Regresa a Roma y haz clic en el botón "Guardar configuración personal". El navegador debería descargar un fichero XML con el mismo nombre de tu esquema. Guárdalo en tu carpeta de trabajo.