TRABAJO PRACTICO ESPECIAL XML

Grupo 01

12 DE NOVIEMBRE DE 2021

Santiago Ballerini, Gonzalo Martone, Agustin Zakalik

Introducción

El objetivo del trabajo práctico consiste en desarrollar un sistema que muestre diversos contenidos de vuelos actuales, utilizando las herramientas de consulta y transformación de documentos XML vistas durante la cursada de la materia.

Toda la información descargada proviene del uso de las API REST de la página web *airlabs.co* . Luego de llamar al API, se generarán tres archivos XML con información de países, aeropuertos y vuelos, la cual luego de ser procesada con el uso de xQuery y XSLT se desplegará en un archivo de LaTeX, que cumpla con la estructura pedida. Para procesar el xQuery y el XSLT se utilizó Saxon en una consola.

Desarrollo

El trabajo comenzó en la página web *airlabs.co* , en la cual fue necesario abrir una cuenta para poder acceder a la API y así descargar la información pedida. Una vez completado este paso, se escribió un bloque de código dentro del archivo tpe.sh con el fin de que descargue los datos necesarios de dicha página, que se almacenaron en los archivos airports.xml, flights.xml y countries.xml.

A continuación, utilizamos una consulta xQuery con el objetivo de unificar la información necesaria en un único archivo XML bien formado. De esta manera, conseguimos centralizar toda la información requerida para la correcta resolución del TPE. Una vez recompilada la información, utilizando XSLT, procesamos el archivo generado por nuestra consulta de xQuery para así obtener un archivo de LaTeX con el formato pedido por la cátedra.

En cuanto a los problemas encontrados, el primero que encontramos fue que no sabíamos cómo verificar la existencia de un nodo en los archivos XML recibidos mediante el API. La siguiente complicación, luego de tener prácticamente toda la estructura básica del xQuery creada, fue cómo simplificar el código. Más específicamente hablando, presentaba código repetido la sección en la cual se generaba el <departure_airport/> y <arrival_airport/>, debido a esto decidimos averiguar si existía la posibilidad de retornar dos nodos, dado que <country/> y <name/> eran los componentes de tanto departure como arrival.

Sin embargo, aunque pudimos solucionar los problemas anteriores, no encontramos una manera de que una función reciba el nombre de un nodo para así simplificar aún más la generación de los distintos aeropuertos. Pero finalmente, sí logramos encontrar una manera de conseguir que la función reconozca si el nodo recibido era un arr_iata o dep_iata, por lo que pudimos obtener el comportamiento deseado, ya que usar el *typeswitch* nos permitió simplificar la generación de un nodo aeropuerto, ya sea departure o arrival.

```
declare function local:buildAirport($iata as element()) as node()* {
    let $airport:= (doc("airports.xml")/root/response/response[./iata_code = $iata])[1]
    let $countryName:= (doc("countries.xml")/root/response/response[./code = $airport/country_code]/name)[1]
    return
        if($airport)
        then
            typeswitch ($iata)
                case element(arr_iata) return
                    <arrival_airport>
                        <country>{$countryName/text()}</country>
                        {$airport/name}
                    </arrival_airport>
                case element(dep_iata) return
                    <departure_airport>
                        <country>{$countryName/text()}</country>
                        {$airport/name}
                    </departure_airport>
                default return ()
        else()
};
```

En cuanto lo que concierne a xQuery, la última complicación que encontramos fue el hecho de que no sabíamos cómo recibir variables externas que nos ayuden a modificar el comportamiento interno, específicamente cómo recibir el valor de error del tpe.sh con el fin de crear el nodo de error en conjunto con su descripción.

En la parte de XSLT tuvimos un problema similar en cuanto a las variables externas que debían indicar la cantidad de vuelos a ser procesados. En este caso afortunadamente la resolución fue más rápida.

Finalmente, en cuanto a los problemas relacionados con la transformación LaTeX, desconocíamos como hacer que el XSLT devuelva &, los cuales son cruciales para la separación de elementos dentro de una tabla en LaTeX. Además, el mayor problema que tuvimos fue el formateado de la tabla para que la misma se vea lo mejor posible. Para esto último, simplemente nos dedicamos a modificar algunos parámetros en el editor online de LaTeX Overleaf hasta que el resultado nos pareció acorde. Una inconveniencia menor que encontramos fue que uno de nuestros errores utilizaba el carácter guión bajo, el cual es un carácter reservado en LaTeX, pero esto lo resolvimos con la sentencia *verb*|**texto-a-reemplazar**|, para que LaTeX no interprete lo que está encerrado entre los pipes.

Los roles asignados para el trabajo fueron:

- Gonzalo Martone: elaboración de la consulta xQuery, co-armado del bash y responsable del funcionamiento global del proyecto.
- Agustin Zakalik: elaboración del reporte y coarmado del bash.
- Santiago Ballerini: elaboración de la consulta XSLT.

Conclusión

El trabajo práctico resultó muy nutritivo en cuanto a la incorporación de conocimientos nuevos no enseñados en la teórica, así como en el ensamblaje de un documento final que requirió integrar los conocimientos adquiridos en clase de distintos lenguajes de consulta.

Aparte de asentar los conocimientos previos que teníamos sobre los lenguajes utilizados, aprendimos a utilizar tanto LaTeX como una API, herramientas que pueden resultarnos muy útiles para trabajos futuros.