EJERCICIO 3

El programa en cuestión es un script de Python por lo que se ejecuta desde la línea de comando. La versión de Python utilizada es la 3.10.5.

El parseado del xml se realiza en un script aparte dentro de la carpeta parse y luego el script lo importa. Esto fue una decisión para optimizar un poco los scripts ya que se iba a repetir código contando con los ejercicios posteriores a este.

Para ejecutarlo es tan sencillo como ejecutar el comando "py .\epubreader.py". (Asegurarse de que se está ejecutando desde la misma carpeta). Después de esto el programa nos preguntará que archivo deseamos parsear, a lo cual pondremos la ruta al archivo deseado (mejor si está dentro de la misma carpeta). El archivo es de tipo epub, los archivos epub son como un archivo comprimido, al descomprimirlo podemos ver que tiene una estructura de directorios. Para nosotros el interesante es el EPUB, dentro del el cual se encuentra el archivo objetivo de este ejercicio, el package.opf, este continene todo tipo de información acerca del archivo, como metadatos sobre el autor, contribuidores, identificador, etc. Nosotros nos vamos a centrar en los metadatos, como poder verlos y como editarlos.

Para ello empezaremos la ejecución del programa. Como todo archivo Python todo empieza de la misma manera.

```
PS C:\Users\jesus\OneDrive\Escritorio\Workspaces\SEW\xml\ejercicio3> py .\epubreader.py Escribe el nombre del archivo epubexample.epub
```

Después de eso se nos pedirá que introduzcamos el nombre del archivo que queremos editar, es recomendable que estén en el mismo directorio para evitar problemas de que no se encuentre el archivo. A continuación, el programa nos enseñara los parámetros que podremos editar

```
Que quieres editar: (En caso de que existan muchos contribuidores se te pedira el número del que quieres editar)
identifier
title
language
date
meta
creator
contributor
```

Aunque se enseñen varios contributor no tenemos por que preocuparnos, ya que como indica la entrada, se le preguntara que contributor se quiere editar más tarde. Aquí podemos equivocarnos sin miedo ya que nos remitirá al mismo punto

```
fajsklfadjslk
Valor no admitido, por favor introduzca un valor correcto
```

Cuando introduzcamos un valor valido en este caso, contributor se nos pedirá que introduzcamos la posición del contributor.

```
Que numero quieres editar: 0-5
6
El valor tiene que ser del 0 al 5 pruebe otra vez.
```

De nuevo los valores de entrada esta controlados para que no salte ningún error inesperado, una vez que escojamos un valor valido se mostrará el valor que tiene dicho elemento,

```
El valor tiene que ser del 0 al 5 pruebe otra vez.
1
Valor de contributor es David Futato
Que valor quieres que tome:
```

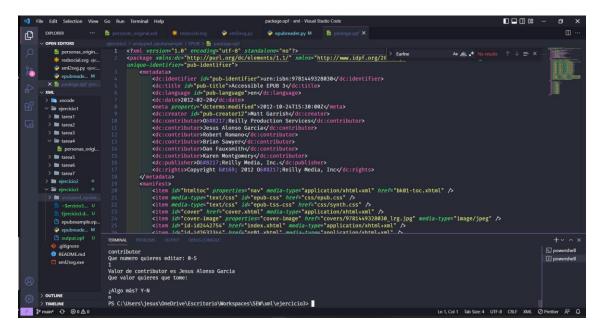
Si no queremos editar el valor tan solo tenemos que darle a enter y ese elemento quedara sin modificar, pero para modificarlo tendremos que introducir un dato.

```
Valor de contributor es David Futato
Que valor quieres que tome:
Jesus Alonso Garcia
¿Algo más? Y-N
```

A partir de ahí podemos seguir editando mas valores o terminar la ejecución del programa una vez fuera del bucle de ejecución el programa terminaría, generando dos salidas, una que queda en el mismo directorio del script, para que se vea la salida y otra que sustituirá al archivo en cuestión.

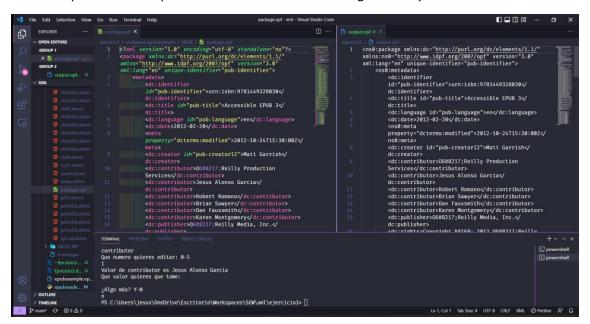
```
| Process | Proc
```

Aquí se puede ver el archivo durante la ejecución del programa, pero cuando salimos del bucle se pueden ver los siguientes cambios



Aquí se puede apreciar tanto la salida generada en el archivo output.opf, como los cambios presentes en el documento editado.

Si bien se pueden ver un par de diferencias entre la salida generada y el archivo editado



Principalmente que se genera sin la cabecera <?xml> y luego también podemos apreciar que se puede ver en cada etiquieta el ns0, esto es debido a que cuando la librería element tree escribe un archivo a partir de su raíz no se reconocen los namespaces previos. Pero esto se ajusta en el archivo editado gracias a la función finetunefile.