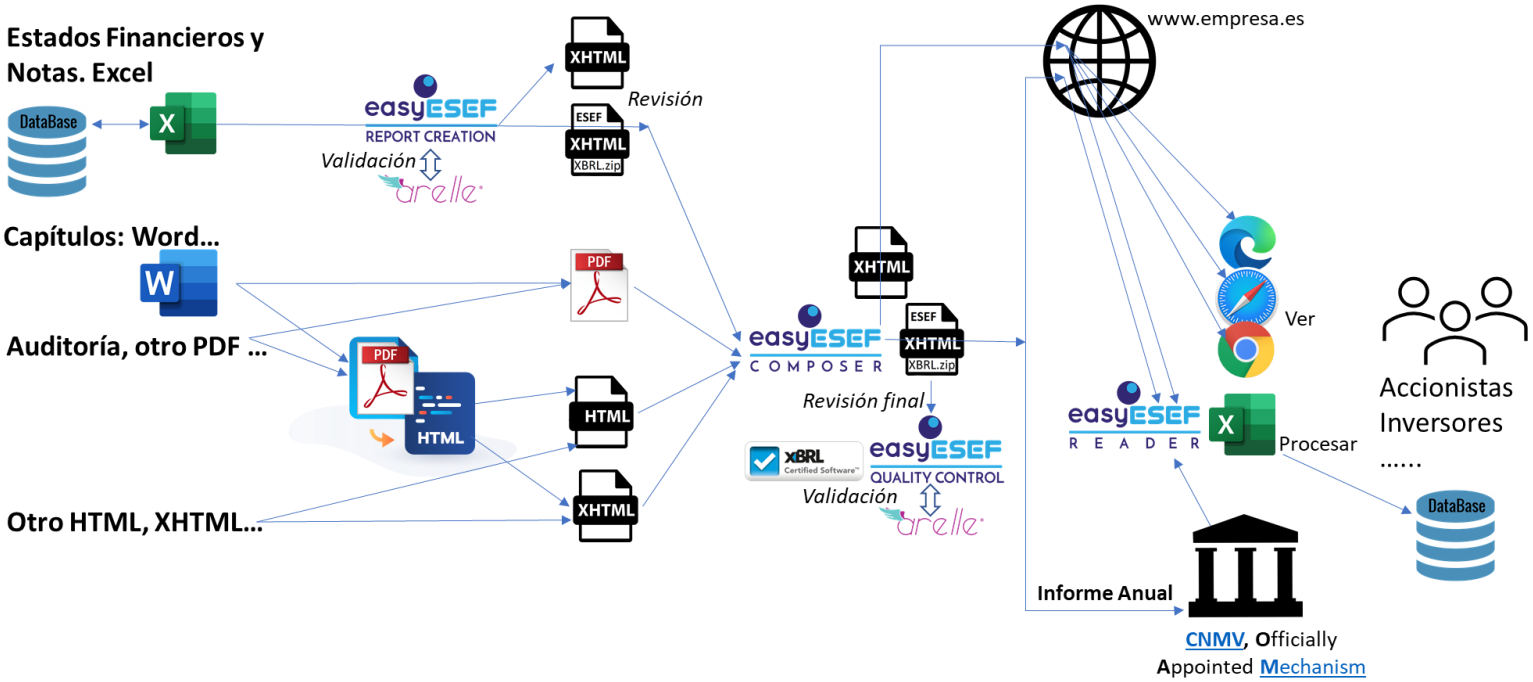
Referencia técnica

easyCOMPOSER: Compone el Informe Anual FEUE

easyCOMPOSER es una herramienta diseñada para exportar un fichero con el Informe Anual en formato FEUE/ESEF. Este fichero es el que se envía a la CNMV y que se publica en la web de la compañía. Importa el fichero .zip con los Estados financieros y Notas ya etiquetados en formato FEUE/ESEF, y los Capítulos y demás componentes en formatos: **.pdf .html .xhtml**



easyCOMPOSER compone el Informe Anual en formato FEUE/ESEF como un fichero con el Informe Anual visualizable en formato .xhtml y ficheros con metainformación obligatoria (*linkbases XBRL*), todo ello comprimido según el estándar *taxonomypackage.zip*

easyCOMPOSER exporta Informe Anual .zip en el directorio que se indique. También exporta una copia del Informe Anual en formato .xhtml para mayor comodidad de uso.

easyCOMPOSER supone que los ficheros importados son correctos. Se recomienda hacer una revisión final con la herramienta ***easyQC*** de Control de Calidad, certificada por XBRL

Internacional. Esta herramienta utiliza internamente el validador Arelle, también certificado por XBRL Internacional.

easyCOMPOSER combina el fichero de Estados financieros y Notas en formato .xhtml (que viene dentro del fichero .zip) con los demás ficheros .html y .xhtml en el orden que se indique. Sólo se puede poner un fichero .zip como máximo. También funciona si sólo se ponen ficheros .html o .xhtml. y se omite el fichero .zip

easyCOMPOSER convierte internamente los ficheros en formato .pdf y .html al formato .xhtml exigido por el formato ESEF. La presentación no varía. Ver detalles en el Anexo *Conversión .html a .xhtml*

easyCOMPOSER detecta internamente si un fichero .xhtml puede afectar a la presentación de otro fichero. Por ejemplo, porque tenga definido un estilo (<*style*>) que aplique a todas las estructuras .xhtml de un tipo determinado (por ejemplo <*table*> o <*td*>). O porque varios ficheros .xhtml utilicen estilos (<*style*>) con el mismo nombre y distintos contenidos. Ver detalles en el Anexo *Desambiguación de estilos .xhtml*

easyCOMPOSER exporta los ficheros .pdf y .html y .xhtml que haya tenido que adaptar al directorio que se indique, para un mayor control.

easyCOMPOSER combina los ficheros origen .xhtml distinguiendo tres partes: <style> <body> y "*otro .xhtml*". El <style> y <body> combinado es la concatenación inalterada de cada <style> y cada <body> de cada fichero .xhtml origen a combinar. Se puede comprobar con cualquier editor de textos que el <style> y <body> del fichero destino combinado no ha variado, que es exactamente la concatenación en secuencia del <style> y del <body> de cada fichero origen combinado. Por su parte, la combinación de "*otro .xhtml*" sigue sus reglas específicas. Ver detalles en el Anexo *Combinar ficheros .xhtml*

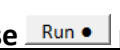
Requisitos de instalación: Microsoft Excel, en versiones Windows o Mac.

Hoja *Composer*

Folder with files: En la celda de debajo se especifica la ruta (*path*) de un directorio por defecto donde:

- Están los ficheros a importar que no tengan su ruta (*path*) especificada
- Se dejan los ficheros exportados
- Se van a crear y borrar los ficheros de trabajo que se necesiten.

Ordered import files (.zip .xhtml .html) to compose: En las celdas de debajo se ponen, por orden, los ficheros .zip .html y .xhtml a combinar. Si un fichero no tiene especificada su ruta (*path*), se utiliza el directorio por defecto.

Pulse  para ejecutar

Composed file exported: Se muestra el fichero .xhtml y (en su caso) el Informe Anual .zip compuesto. Ambos ficheros estarán en el directorio por defecto.

Files with .xhtml adjusts exported: Por cada fichero .xhtml que haya necesitado algún tipo de ajuste, se muestra ese fichero, que estará en el directorio por defecto.

Converter from .pdf to .html: Ruta absoluta, o relativa desde el directorio por defecto, al *conversor.exe* .pdf a .html añadiendo los parámetros oportunos (zoom 1.1 por ejemplo):

pdf2htmlEX-win32-0.14.6-upx-with-poppler-data\pdf2htmlEX.exe --zoom 1.1

Conversor usado: soft.rubypdf.com/software/pdf2htmlex-windows-version Es de código abierto y libre descarga. Es el más utilizado en ESEF, en sus distintas versiones.

Como conversor comercial, se ha probado PDF2HTML Reflow Paragraph Module con buenos resultados. No dispone de la opción zoom. La versión demo se descarga y funciona perfectamente, pero sólo convierte las páginas impares.

Hoja *ListOfFiles*

Folder with files: En la celda de debajo se especifica la ruta (*path*) de un directorio del que se van a mostrar todos los ficheros y subdirectorios que contenga. Es una ayuda de utilidad.

Pulse Run para ejecutar

List of files and folders: Se muestra la lista de ficheros y subdirectorios que contiene el directorio especificado.

Para saber más

Para información más detallada y preguntas sobre ***easyESEF***, por favor diríjase a info@easyESEF.eu

Vea nuestras otras herramientas para ESEF en easyesef.es/resources. Además de productos, tenemos disponible soporte altamente especializado para ESEF. Este servicio es independiente a cualquier proveedor y es aplicable a cualquier software de generación ESEF

FIN DEL TEXTO PRINCIPAL DE LA REFERENCIA TÉCNICA

Anexo: Conversión .html a .xhtml

El formato ESEF requiere que el fichero esté en formato .xhtml Sin embargo, hay veces que el fichero está en formato .html y se precisa una conversión. La conversión se puede comprobar con https://validator.w3.org/

easyCOMPOSER convierte un fichero .html en un fichero .xhtml en los siguientes pasos:

a) Traducir ciertas etiquetas[1] y caracteres especiales[2] de .html a .xhtml
b) Poner en un comentario las instrucciones del tipo como <![if ! IE]> a <!--<![if ! IE]>-->
c) Quitar CR LF TAB, espacios dobles en etiquetas, según https://www.w3.org/TR/xhtml1/guidelines.html C5
d) Los nombres de etiqueta y los atributos en etiqueta que estén con todas las letras en mayúsculas se cambian a minúsculas como COLSPAN=3 a colspan=3
e) Los valores de los atributos en las etiquetas se ponen siempre entre comillas, como colspan=3 a colspan="3"
f) Las etiquetas <img...> y <meta...> que no estén cerradas con </img> o </meta> se cierran siempre como <img... /> y <meta... />.
g) La etiqueta <img > debe tener el atributo alt="...." definido. Incluir alt="" en caso contrario.
h) Las etiquetas que se autocierren deben dejar un espacio justo antes del /> final, como
i) En la etiqueta <html....> si el namespace xmlns="http://www.w3.org/1999/.xhtml" no estuviera, incluirlo, como <html.... xmlns="http://www.w3.org/1999/.xhtml" ...>

Anexo: Desambiguación de estilos .xhtml

easyCOMPOSER revisa cada fichero .xhtml a combinar, en el orden en que se hayan declarado en la hoja *Composer*, para determinar si precisa de ajustes de desambiguación.. Si hay un fichero .zip, este fichero se revisa el primero.

Se ajustarán los estilos de la zona <style> del fichero .xhtml si se detecta:

- Un estilo directamente aplicable a tags .xhtml como table{} td{} que serían directamente aplicables a todos los tags <table> <td> del fichero .xhtml combinado.
- Un nombre de estilo ya utilizado en un fichero anterior .xhtml pero con otro contenido (por ejemplo, con fuente de letra distinta).

En cada fichero a ajustar:

- Se prepara un sufijo único (como "____N") para cada fichero a ajustar.

---

[1]
</br>|
|<hr></hr>|<hr>|<embed |</embed>|<style type=""text/css""> |<del> |</del> |<u> |</u>|<nobr>|</nobr>

|
|<hr />|<hr />|<img |</img>|<style>|<span style=""text-decoration:line-through;"">||<span style=""text-decoration:underline;"">||<span style=""white-space:nowrap;"">|

[2] |"|&|'|<|>
 |"|&|'|<|>

- Se desambigua concatenando a cada nombre de estilo original el sufijo único para crear un nombre de estilo único. Esto elimina la posibilidad de nombres de estilos duplicados. Ejemplo: .ft0 #page_1 cambiados a .ft0____N #page_1____N
- En el caso de estilos directamente aplicable a tags .xhtml como table{} o td{}, se añade un punto "." al principio del nombre del estilo. Ejemplos: table{} td{} cambiados a .table____N {} .td____N {}
- En la zona <body> se cambian los nombres de estilo originales por los nuevos nombres de estilo únicos Ejemplo: <p class="ft0 "> cambiados a <p class="ft0___ N ">
- En la zona <body>, en los tags con estilos directamente aplicable a tags .xhtml como table{} o td{}, se añade el nuevo nombre único de estilo como atributo class="" como en los ejemplos <table > <td > cambiados a <table class=".table___N" > <td class=".td___N" >

Anexo: Combinación de ficheros .xhtml

easyCOMPOSER toma cada fichero origen .xhtml a combinar, en el orden en que se hayan declarado en la hoja *Composer*. De cada fichero, extrae el contenido de <style>PAYLOADSTYLE</style> y <body>PAYLOADBODY</body> así como cierta información de etiquetas de *"otro .xhtml"* como <title>…</title>.

El fichero destino .xhtml es el resultante de concatenar estos contenidos en un solo <style>…</style> y en un solo <body>….</body>, así como de componer la información de las etiquetas de *"otro .xhtml"*

De cada fichero "n" origen a combinar se extraen tres tipos de información:

1. Contenido de (puede haber ninguno, uno o varios) <style>PAYLOADSTYLEn</style>
2. Contenido de <body> PAYLOADBODYn</body>
3. Contenido de "*otro .xhtml*", por ejemplo:
   a) Línea inicial <?xml version="1.0" encoding="UTF-8"?>
   b) Tag <!DOCTYPE > (se ignora)
   c) Namespaces adicionales en <html…>
   d) Tag <title>….</title>
   e) Tags no repetidos diferentes a los anteriores, tomando nota de su posición relativa en la estructura .xhtml

easyCOMPOSER exporta el fichero .xhtml combinado, con una estructura .xhtml de tipo:

```
<?xml…>
<html….>
        <head>
                <title>…</title>. El fichero .zip tiene prioridad aquí
                <meta  /> y otras etiquetas no duplicadas, en la posición que tuvieran
                <style>PAYLOADSTYLE1PAYLOADSTYLE2PAYLOADSTYLE3</style>
        </head>
<body>PAYLOADBODY1PAYLOADBODY2PAYLOADBODY3</body>
</html>
```


Anexo: Códigos hash SHA256 en ficheros .xhtml

Problema práctico.

Una Compañía está preparando su Informe Anual auditado. Cada Capítulo se prepara hipotéticamente por un Departamento diferente. El Auditor va recibiendo los diferentes Capítulos según estén disponibles, para avanzar en paralelo, dada la habitual premura de plazos. Hay que comprobar que el Informe Anual se compone única y exclusivamente de la integridad de los Capítulos auditados.

En el caso del formato electrónico único europeo FEUE/ESEF, el Informe Anual es un fichero visualizable en cualquier navegador Internet (formato .xhtml), más una serie de ficheros no visualizables complementarios a los estados financieros (estándar iXBRL).

Por homogeneidad, los Capítulos se van enviado individualmente al Auditor cómo ficheros visualizables .xhtml. Al final, los Capítulos se combinan creando el Informe Anual .xhtml.

Código hash SHA256 como huella digital

Un código hash es la huella digital del contenido de un fichero. Hay varios algoritmos para obtener el código hash de un fichero. El código *hash* se obtiene mediante el algoritmo de referencia estándar en criptografía SHA256 que produce una huella digital irrepetible de 256 bits[3]. Cualquier alteración del contenido del fichero alterará el código SHA256. El algoritmo SHA256 es de seguridad fuerte, de manera que no se conoce el método para crear un fichero cuyo SHA256 obtenido tenga precisamente un valor conocido. De esta manera se garantiza la integridad: si del fichero se obtiene un SHA256 conocido, es que el fichero no ha sufrido alteración alguna.

El código hash SHA256 no depende del sistema operativo ni de ningún software concreto. El código hash SHA256 depende de la codificación de los caracteres, que será UTF8 de acuerdo con el formato ESEF. Obtener el SHA256 de un fichero o tira de caracteres es fácil con librerías de programas y en soluciones online.

Por tanto, conocidos los códigos hash SHA256 de una serie de ficheros, siempre se puede comprobar que esos ficheros no han sufrido alteraciones.

Concatenar varios ficheros origen en un solo fichero destino no altera el código hash SHA256 de los ficheros concatenados. Si se conoce el tamaño de cada fichero origen a concatenar, y la secuencia de concatenación, se puede extraer uno a uno (“desconcatenar”) cada fichero origen de la concatenación y volverle a calcular su código hash SHA256.

[3] Estos 256 bits se suelen representar o bien como 64 caracteres hexadecimales (0-9:A-F, base 16) o bien como 44 caracteres A-Z:a-z:0-9:+:/ en base 64 y con un = al final.

Composición de ficheros .xhtml

Un fichero .xhtml se puede describir básicamente como tres tipos de estructuras xml:

- <body>….</body> que contiene los textos e imágenes a representar, y la estructura de cómo representalos (párrafos, tablas, listas…)
- <style>….</style> que contiene los estilos aplicables a <body>….</body>, como fuentes y tamaños de letra, ciertas imágenes repetitivas, colores, alineaciones …
- *"otro .xhtml"* como <title>…</title>, espacios de nombres (namespaces) y otra metainformación.

Al combinar varios ficheros .xhtml origen en un fichero .xhtml destino, por una parte se combina la zona <body>….</body> de cada fichero .xhtml origen, para dar lugar a un solo <body>….</body> en el fichero .xhtml destino, ya que sólo puede existir una única estructura <body>….</body> en un fichero .xhtml

Una combinación similar se hace para la zona <style>. Y ciertas partes duplicadas de cada fichero origen (como las propias etiquetas <body></body>) simplemente desaparecen por redundantes.

Cada fichero .xhtml se ha descompuesto en diversas zonas, que se han recombinado independientemente en el fichero .xhtml destino. Ya no se puede volver a calcular el código hash SA256 de cada fichero .xhtml origen observando al fichero .xhtml destino.

PROBLEMA: ¿Cómo garantizar la integridad de cada fichero .xhtml o .html origen cuando se combinan en un sólo fichero .xhtml destino?

Composición de ficheros origen .xhtml con <style> y <body> invariante

Un procedimiento sencillo de obtener un fichero .xhtml destino es concatenar el contenido de la estructura <body>PAYLOADx</body> de todos y cada uno de los ficheros .xhtml origen en una sola estructura <body>PAYLOAD1PAYLOAD2PAYLOADn</body> en el fichero .xhtml destino. Cada fichero .xhtml tiene una y sólo una estructura <body>PAYLOADx</body>

Se guarda el tamaño y el SHA256 del PAYLOADx de cada fichero .xhtml origen.

Sabiendo el tamaño de cada PAYLOADx, y sabiendo el orden en que están concatenados (puestos uno justo a continuación de otros), se vuelve a calcular el SHA256 de cada PAYLOADx en la estructura <body>PAYLOAD1PAYLOAD2PAYLOADn</body> del fichero .xhtml destino.

Si los SHA256 calculados coinciden con los SHA256 guardados, se demuestra entonces que los PAYLOADx respectivos no han variado.

Se puede seguir la misma mecánica para las estructuras <style>PAYLOADx</style>. Cada fichero .xhtml origen tiene cero, una o varias estructuras <style>PAYLOADxy</style>. Primero se concatenan todas las estructuras <style>PAYLOADxy</style> en cada fichero .xhtml origen como si fueran una sola estructura <style>PAYLOADx</style> (que puede estar vacía). Se guarda su tamaño y su SHA256 (si el tamaño es mayor que cero). Y se concatena

en la estructura <style>PAYLOAD1PAYLOAD2PAYLOADn</style> del fichero .xhtml destino. Conociendo los tamaños, se pueden volver a calcular los SHA256 y comprobar que el contenido del fichero .xhtml destino es invariante respecto a la concatenación de los ficheros .xhtml origen.

Como la códigos presentación .xhtml se hace a partir de textos e imágenes (zona <body>) y de estilos a aplicar (zona <style>), con este mecanismo de hash SHA256 se puede comprobar que los ficheros .xhtml origen están combinados en el fichero .xhtml destino sin alteración alguna.

Composición de ficheros origen .xhtml con <style> y <body> no invariante: "distilled"

Pueden darse casos en que haya variaciones las zonas <style>PAYLOADx</style> y/o <body>PAYLOADx</body> de un fichero .xhtml origen.

Por ejemplo que sea un fichero .html que haya que transformar a .xhtml (ver el Capítulo *Conversión .html a .xhtml*). O que ciertas definiciones de estilos en el fichero afecten a otros ficheros .xhtml origen (ver el Capítulo *Desambiguación de estilos .xhtml*).

En estos casos, puede haber multitud de cambios dentro de las estructuras xml, por ejemplo si se le cambia el nombre a un estilo para desambiguarlo, o si se ponen los valores de los atributos entre comillas según el estándar .xhtml

Si las secuencias de caracteres en las estructuras xml varían, los tamaños varían y los códigos hash SHA256 varían.

Lo que no varían son los textos ni las imágenes. Las transformaciones en la metainformación .xhtml (como se han de representar los textos e imágenes) no alteran lo que es la información en sí de textos e imágenes (sus valores). Donde hay un 3, no va a aparecer un 4. Y donde hay una imagen de un coche, no va a aparecer la imagen de un avión.

Sin embargo, comprobar que no haya habido cambios en textos e imágenes con una comparación clásica carácter a carácter (como comparar documentos en Word) detecta tantas diferencias que no es de utilidad.

La solución propuesta es ignorar las estructuras .xhtml y tomar como invariantes únicamente texto e imágenes.

Aunque cambien los estilos de visualización o partes del etiquetado del fichero .xhtml o .html no han de cambiar ni textos ni imágenes. Y si texto e imágenes no varían, se pueden "destilar" (extraer en orden) texto e imágenes desde la estructura <body>PAYLOAD</body> de un fichero .xhtml (o .html) a un fichero "destilado" de trabajo.

Se guardan los tamaños y SHA256 del fichero "destilado" correspondiente a cada fichero .xhtml o .html origen.

El fichero .xhtml destino se crea por concatenación de las estructuras <body></body> de los ficheros origen. El fichero "destilado" destino resultante será a su vez la concatenación de los ficheros "destilados" origen, pues ni texto ni imágenes varían en el proceso de combinación.

Conociendo el tamaño y SHA256 de cada fichero “destilado” origen, se puede comprobar que son invariantes en el fichero ”destilado” destino resultante.

Una vez que se tiene el Informe Anual, su contenido “destilado” se trocea según el tamaño “destilado” de cada Capítulo que lo compone. El SHA256 de cada trozo tiene que coincidir con el SHA256 del “destilado” del Capítulo respectivo. De esta manera se garantiza que el Informe Anual .xhtml es precisamente la concatenación de los Capítulos .xhtml o .html que lo componen, independientemente de los cambios en estructuras xml que pudiera haber habido. Cualquier alteración hará que los SHA256 dejen de coincidir.

En teoría se abren posibilidades de alterar la visualización de tal manera que parezca que el contenido es otro. Puede ser un interesante tema de investigación académica, pero es poco práctico, pues la manipulación sería de efectos limitados y demasiado notoria.

UNA SOLUCIÓN PRÁCTICA:

easyCOMPOSER calcula los tamaños y códigos hash SHA256 del <style>, <body> y *“destilado”* (sólo texto e imágenes) de cada fichero origen a combinar, y también del fichero destino .xhtml combinado. Se pone esta información como un comentario no visible al final del fichero destino combinado. El código hash SHA256 del <style> y <body> de cada fichero origen .xhtml a combinar es idéntico al código hash SHA256 de la zona correspondiente en el <style> y <body> del fichero destino combinado. El código hash SHA256 del *“destilado”* de cada fichero .xhtml o .html a combinar es idéntico al código hash SHA256 del *“destilado”* del fichero combinado, aunque al combinar se hayan modificado etiquetas o atributos en <style> y/o <body>. Ver detalles en el Capítulo *Códigos hash SHA256 en ficheros .xhtml*

easyCOMPOSER calcula y muestra los tamaños y códigos hash SHA256 del <style>, <body> y *“destilado”* de cada fichero que se indique en la hoja *HashCodes* y comprueba que coincidan si es un fichero combinado. La utilidad es que se puede auditar en paralelo Capítulo a Capítulo en formato .xhtml con garantías de integridad. Se van guardando los códigos hash SHA256 de cada Capítulo y se puede comprobar que sus códigos hash SHA256 no hayan variado en el Informe Anual. De hecho, el SHA256 del *“destilado”* (sólo texto e imágenes) no varía aunque ***easyCOMPOSER*** haya pasado el fichero de .html a .xhtml o haya tenido que ajustar estilos.

easyCOMPOSER calcula cada vez cada código hash SHA256 y comprueba (si hay suficiente información) que coincida con los declarado en la metainformación del fichero compuesto .xhtml poniendo un ***OK*** a la derecha del SHA256. Si no coincide, muestra el SHA256 discrepante o el error detectado.

Hoja *HashCodes*

Folder with files: En la celda de debajo se especifica la ruta (*path*) de un directorio por defecto donde:

- Están los ficheros y subdirectorios que no tengan su ruta (*path*) especificada
- Se van a crear y borrar los ficheros de trabajo que se necesiten.

List of files (and folders) for hash codes: En las celdas de debajo se ponen los ficheros de los cuales se van a mostrar los códigos hash SHA256. Si se pone un subdirectorio, se procesarán todos los ficheros que contenga.

Pulse Run para ejecutar

File name: Nombre del fichero o subdirectorio procesado.

- Si es un fichero .zip o un directorio, se expande y se procesan todos los ficheros y directorios que contenga.
- Se utiliza el sangrado (poner el nombre del fichero hacia la derecha) para mostrar el orden jerárquico dentro del fichero o directorio.
- Si es un fichero .xhtml compuesto, las líneas siguientes tienen el nombre entre paréntesis de cada fichero que forma parte de la composición.
    - El primer fichero es el propio fichero compuesto, pero sin la metainformación de composición (unos 500 bytes).
    - Los siguientes son cada fichero origen que componen el fichero compuesto.

File size: Tamaño del fichero en bytes.

File SHA256: Código hash SHA256 del conjunto de todos los bytes del fichero.

Body size: Tamaño de la zona <body> en bytes.

Body SHA256: Código hash SHA256 del conjunto de todos los bytes de la zona <body>.

Style size: Tamaño de la zona <style> en bytes.

Style SHA256: Código hash SHA256 del conjunto de todos los bytes de la zona <style>.

Distilled size: Tamaño de la zona <body> "*destilada"* (sólo texto e imágenes) en bytes.

Distilled SHA256: Código hash SHA256 del conjunto de todos los bytes de la zona <body> *"destilada"* (sólo texto e imágenes).

☑ ***Check for export distilled as file .distilled.txt*** Si está activado, se exporta un fichero conteniendo la zona <body> *"destilada"* (sólo texto e imágenes) del fichero. El fichero se exporta en el directorio por defecto, y mantendrá el mismo nombre de fichero añadiendo al final "*.distilled.txt*"

Algoritmo de destilado *"distilled"* (textos e imágenes sólo) de un fichero .xhtml o .html:

1. Borrar desde el principio del fichero hasta la etiqueta ***<body ….>***
2. Borrar desde la etiqueta ***</body>*** hasta el final del fichero
3. Borrar todos los comentarios ***<!-- … -->***
4. Reemplazar cada imagen ****** por sólo ***data:image/……***
5. Borrar todas las etiquetas XML ***<etiqueta….>*** así como ***</etiqueta>***
6. Borrar todas las secuencias de escape de caracteres ***&xxx;*** como   o
7. Borrar retornos de carro, saltos de línea, tabuladores y espacios.

Al final queda un contenido *"distilled"* invariante, algo así como:

Memoriadelejercicio2020data:image/OrK….5CYII=CuentasAnualesdelejercicio…..

El algoritmio de *"destilado"* puede programarse en cualquier lenguaje o con expresiones regulares. La codificación de caracteres es UTF8 según el formato ESEF.

Guardar tamaños y códigos hash SHA256

Se recomienda utilizar la siguiente estructura de comentario xhtml para guardar la información de tamaños y códigos hash SHA256. Esta estructura se repite por cada fichero origen y para el propio fichero destino. Estas estructuras se colocan después de </body> en el fichero destino.

<!--hashfile order="" file=""
filesize="" filehash=""
bodysize="" bodyhash=""
stylesize="" stylehash=""
distilledsize="" distilledhash="" -->

order="" Obligatorio. Secuencia de dígitos 0..9 no repetida. Indica el orden de este fichero origen dentro del fichero destino. order="0" indica fichero destino.

file="" Obligatorio. Nombre del fichero, sin ruta (path)

filesize="" Opcional. Tamaño del fichero. En el caso de order=”0” tamaño del fichero hasta </body> inclusive.

filehash="" Opcional. Sólo si filesize > 0. Código hash SHA256 de ese tamaño del fichero (en el caso de order=”0” tamaño del fichero hasta la etiqueta </body> inclusive.

bodysize="" Opcional. Tamaño del PAYLOAD en la estructura <body...>PAYLOAD</body> del fichero.

bodyhash="" Opcional. Sólo si bodysize > 0. Código hash SHA256 del PAYLOAD en la estructura <body...>PAYLOAD</body> del fichero.

stylesize="" Opcional. Tamaño del PAYLOAD en la estructura <style...>PAYLOAD</style> del fichero, concatenado si hubiera varias.

stylehash="" Opcional. Sólo si stylesize > 0. Código hash SHA256 del PAYLOAD en la estructura <style...>PAYLOAD</style> del fichero, concatenando si hubiera varias.

distilledsize="" Opcional. Tamaño del "destilado" del PAYLOAD en la estructura <body...>PAYLOAD</body> del fichero.

distilledhash="" Opcional. Sólo si distilledsize > 0. Código hash SHA256 del "destilado" del PAYLOAD en la estructura <body...>PAYLOAD</body> del fichero.

Ejemplo:

```
</body><!--hashfile order="0" file="company-2021-12-31.xhtml"
filesize="2051206" filehash="TockHgZ0BgsQCgLEx13ZsWdNLCqlDmFKC5rwR6SBVyQ="
bodysize="2042432" bodyhash="zxm9BacGlski1dbp+DfQ/zVuj2agm9rmaVIXu9dI36I="
stylesize="5580" stylehash="Bw+W0qjof9UuTfJasOUOpNV32d7ZBXRbotth+uqtTSA="
distilledsize="133796"
distilledhash="jmc62732j6gzjXcA8GvgEn7mTRghJmLMQYby2ZW8Q2U="      -->
```

Prueba preconstituida:

Una forma sencilla de guardar indeleblemente[4] los SHA256 es enviarlos como mensaje por Whatsapp, Instagram, Twitter u otro servicio seguro de mensajería. Tanto remitente como destinatario (o cualquiera otra persona que tuviera acceso) podría usar ese mensaje como prueba judicial preconstituida.

[4] Puede usarse también firma electrónica, o soluciones avanzadas como sigstore.dev (firma software transparente) u openfiling.info/blockchain (firma blockchain, del mismo autor de este documento)

