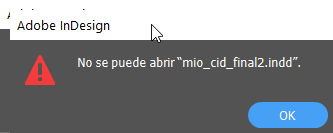
## 1. El problema

Supongamos que diez años atrás editamos y publicamos un libro, que ahora necesitamos reimprimir. Los archivos siguen almacenados en un disco duro, de modo que bastaría recuperarlos, abrirlos en InDesign, hacer las correcciones y ajustes que hiciesen falta, y volver a imprimir.

En el disco duro encontramos la carpeta con el proyecto y dentro de esta carpeta, una subcarpeta llamada “tripa” donde efectivamente hay un archivo InDesign. Perfecto. El problema es que mi versión de InDesign decide que no es un archivo con quién quiera tener algún tipo de trato.



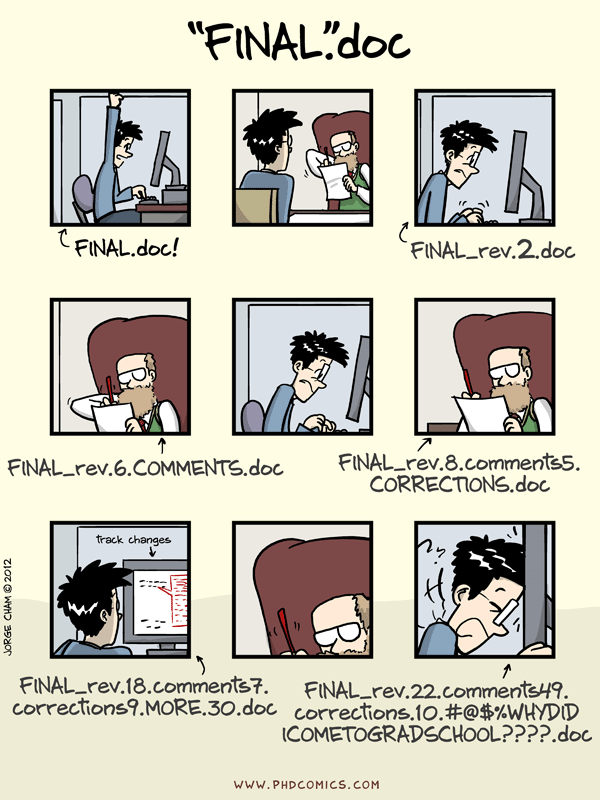
InDesign no puede abrir el archivo

El mensaje no solo es desalentador, también es totalmente opaco. ¿Por qué no se puede abrir? ¿El archivo está corrupto? ¿Tiene que ver con la versión de InDesign que estoy usando? Ninguna pista. Podría probar utilizando la versión de InDesign con la que cree el archivo, pero, demonios, hace diez años de eso, ¿cuál podría haber sido? Sin contar con que tendría que descargarla e instalarla y sin ninguna garantía de que el asunto funcionase.

Vamos, pues, al plan B. Que no es otro que localizar la última versión del manuscrito en MS en Word y rehacer la maqueta.

El manuscrito en cuestión se produjo después de un laborioso escaneo: nada extraño considerando que la última edición del libro en cuestión era de hace unos treinta o cuarenta años atrás. Cualquiera que halla escaneado un libro sabe que los resultados distan mucho de definir la palabra prolijidad. El resultado: la corrección tomó un tiempo, primero utilizando herramientas automatizadas (grep, fundamentalmente, pero ya hablaremos de eso), y un trabajo denodado de nuestro corrector. Por tanto, el ir y venir de borradores por correo electrónico fue profuso.

Considerando lo anterior, resulta natural (preocupante, pero natural) que cuando miramos en la carpeta que contiene los borradores, encontremos no uno sino una serie de archivos (versión2, versiónfinal, versión definitiva… espera, ¿“versiondefinitiva” es más final que “final” ¿o es al revés?). Siempre podemos buscar sencillamente la última versión del documento en MS Word y trabajar con esa. Pero, ¿es realmente la última o la corrección final fue hecha en InDesign? De ser así, posiblemente podríamos recuperar el log del control de cambios en InDesign… solo que no podemos. ¿O tal vez la revisión más reciente incluía cosas que descartamos en la versión final del libro? Es una posibilidad. De paso y ya que estamos, convendría también revisar los correos electrónicos con el corrector.



documento “final”

Al final escogemos un archivo, y lo abrimos. Naturalmente, los encabezados brillan por su ausencia y después de echar un vistazo en el panel de estilos, sabemos que mapear esto e importarlo de alguna manera indolora en InDesign va a ser la más utópica de las proezas.

Llegado a este punto, habrán pasado un par de horas y estaremos sentados frente al ordenador con deseos de tirarlo por la ventana y correr a ahogarnos de chupitos en algún antro infame.

### 1.1

Cosas como estas pasan todo el tiempo. Sean estos los problemas, u otros distintos, o una combinación variable de problemas conocidos y otros totalmente novedosos, la combinación MS Word + InDesign rara vez es indolora, sobre todo si consideramos documentos enviados por correctores o traductores o escritores: pueden lucir muy bien, pero rara vez están bien estructurados. Y la tentación del diseño (que es una especie de enfermedad que se apodera tarde o temprano de los usuarios de Word) termina por crear monstruos, para mayor horror de diseñadores y editores. Discutiremos esto en otro lugar, pero sí podemos mencionar rápidamente los problemas de MS Word:

* Nada sostenible (abrir un documento de una versión antigua de word y esperar que todo funcione es una operación que involucra una dosis importante de azar)
* El código que genera es confuso y pesado
* Su sistema de control de versiones es limitado (en funcionalidad y alcance)

Y sin embargo, word es el estándar, lo cual no deja de resultar misterioso. ¿No hay otra manera de hacer las cosas? En efecto la hay. Pero aunque es más eficiente y fluida y nos abre numerosas posibilidades, por alguna razón nadie o casi nadie la utiliza. Ese es el camino que vamos a explorar. Lo que voy a proponer aquí es un flujo de trabajo digital, open source y sostenible, que en mi opinión resuelve muchos de los problemas del workflow tradicional, cuyo principal defecto es que está orientado primariamente a la producción de un material impreso en un entorno digital. Con esto, desde luego, no quiero insinuar que tengo algo en contra del libro impreso, sino más bien que es posible *también* hacer libros impresos bellamente formateados partiendo de un concepto de producción puramente digital.

## 2. Objetivo, herramientas & requisitos

### Objetivo

El propósito de este tutorial será mostrar desde un punto de vista práctico como combinar herramientas para crear un libro multiformato, de una mera sostenible y automatizada, y que, de paso, resuelve punto por punto nuestro problema..

Este es únicamente un posible *workflow* y desde luego que existen muchas otras herramientas y combinaciones. El punto que espero demostrar es únicamente la cantidad de herramientas que existen a nuestra disposición, que pueden transformar el proceso de producción de un libro hasta el punto de permitirnos automatizar todas las tareas, ganando en eficiencia y al mismo tiempo en control sobre nuestros proyectos.

Para mostrar esto, voy a replantear mi proyecto (la edición de *Mío cid campeador* de Vicente Huidobro) y rehacerlo desde el principio. Para ello:

* Crearemos, a partir de los archivos que tenemos, un set de archivos fuente en markDown (texto plano con la extensión .md)
* Luego crearemos un repositorio para albergar los archivos necesarios
* Crearemos un archivo formateado en yaml para almacenar la metadata del libro
* Pondremos este repositorio (el directorio donde vive nuestro proyecto) bajo control de versiones con git
* Sincronizaremos este repositorio local con un repositorio remoto en GitHub
* Utilizando pandoc, crearemos nuestro Epub, y crearemos un archivo ICML para importarlo en inDesign
* Crearemos desde InDesign el PDF final de imprenta
* Veremos como hacer correcciones, manteniendo un único set de archivos fuente y guardando un registro histórico de correcciones.
* y por último, crearemos una página web con los archivos de nuestro libro.

### 2.1 Requisitos

Para seguir este tutorial, es aconsejable (aunque no indispensable):

* Alguna familiaridad con la línea de comandos.
* Tener conocimientos básicos de HTML y CSS.
* Un conocimientos de InDesign a nivel de diseño editorial.
* Una mente abierta (todo lo anterior se puede reemplazar con una aplicada búsqueda en Google, esto último no).

### 2.3 Herramientas & software

Y deberemos tener instalado en nuestro PC:

* Un editor de texto plano (no procesador de texto): yo utilizaré Atom, pero hay una serie de editores de texto igualmente útiles y libres (Visual Studio Code, Sublime Text, Vim, etc…)
* Línea de comandos (en Windows, el PowerShell; en Linux, el terminal o uno cualquiera de sus emuladores; en Mac, la terminal).
* Pandoc
* InDesign
* Git y el cliente de escritorio de GitHub
* MS Word
* Jeckyll

Instalar software es tedioso. De momento, vamos a sumir que (como yo) tenéis todo el software instalado en vuestro ordenador.

Y dado que de lo que se trata es de probar algo distinto (no se puede esperar un resultado diferente si hacemos lo mismo), vamos a cambiar nuestro enfoque y utilizar Markdown como formato base, en lugar del consabido MS Word. Las razones para escoger markdown son exactamente las mismas que para desechar MS Word y se pueden resumir en los tres mantras de cualquier workflow editorial en un entorno digital:

* estructura
* sostenibilidad
* interoperatividad

Necesitamos un archivo bien estructurado. Esto garantiza que pueda ser mapeado correctamente para la importación en InDesign, que se pueda generar un epub automáticamente, y que este epub, tal cual ha sido exportado por pandoc, sea accesible.

El hecho de que trabajemos en texto plano (markdown más que un formato, es una convención para etiquetar texto plano), garantiza que estos archivos sean legibles por cualquier ordenador, sea cual sea el sistema operativo y el software que tengan instalado, ahora, hace treinta años o en el siglo XXII. Además, son tan interpretables por máquinas como legibles por humanos, y pesan ridículamente poco comparados con sus colegas con ínfulas, los archivos generados por procesadores de texto como MS Word o Open Office.

Y por estas razones, son también fácilmente trasladables (de una maquina a otra, de un software a otro, de un lenguaje de etiquetado a otro). Si el archivo está bien estructurado, podemos hacer en efecto lo que queramos con él.

Manos a la obra.

## Configurar el espacio de trabajo

### El espacio de trabajo

Lo primero será crear el directorio donde van a vivir los archivos de mi proyecto. En mi ordenador será la carpeta mioCidCampeador. Dentro de esta carpeta, además crearé una carpeta para las imágenes, otra para los archivos markdown y una última que llamaré src (por “source”: el archivo fuente o archivo base). para ello, nos situamos con el terminal en el directorio donde queremos crear la carpeta de nuestro proyecto), y escribimos:

mkdir mioCidCampeador  
cd mioCidCampeador  
mkdir imagenes, markdown, src, indd, icml,

Naturalmente, también se puede crear la carpeta haciendo click derecho > crear carpeta, pero para los propósitos de este tutorial, utilizaremos la línea de comandos: es más rápido y de esta manera nos familiarizamos con su uso.

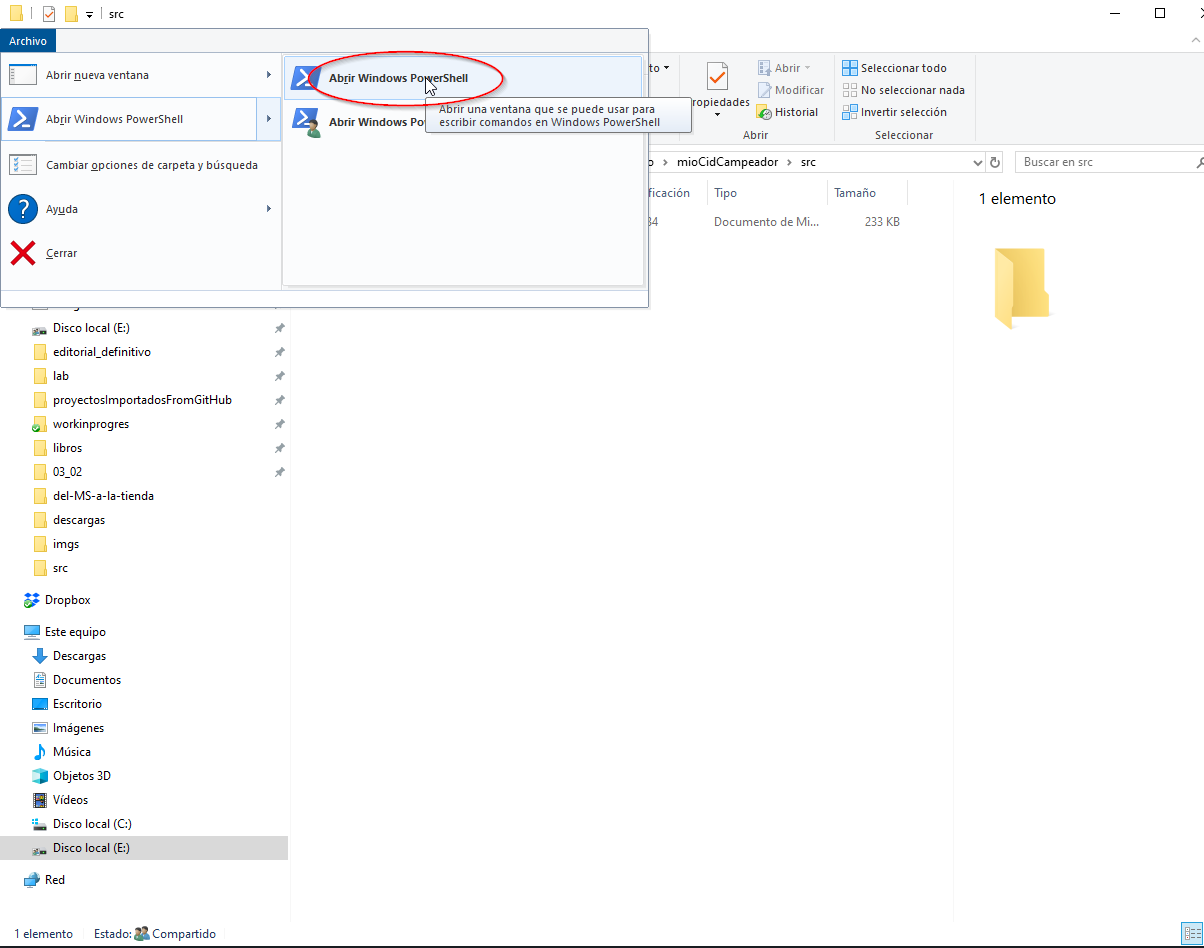
Una vez creada nuestra carpeta, podemos poner nuestro documento (si: mioCidFinal.docx) en la carpeta correspondiente (src). Este es el punto de partida de nuestro proyecto.

## Formatear

### Preparar el MS

De momento tenemos una carpeta, una serie de subcarpetas, y un único archivo. Pero, ¿no íbamos a partir desde markdown? En efecto. Por tanto, lo primero que haremos será convertir el documento base (mioCidFinal.docx) a markdown. Y aquí es dónde entra Pandoc.

Pandoc es un software de linea de comandos (CLI), creado y mantenido por John MacFarlane, profesor de Filosofía en la Universidad de California en Berkeley. Muy brevemente, se puede describir como un conversor universal de formatos de texto. Pero es mucho más. Y puesto que fue concebido por un académico y con el objetivo de servir de herramientas a académicos, cuenta entre sus virtudes la capacidad de manejar metadata, referencias bibliograficas y sistemas de citación que mejoran casi cualquier herramienta dedicada a estas tareas. Hablaremos más adelante más en detalles de qué hace y cómo lo hace, de momento (y como demostración de lo que es capaz), vamos a navegar a la carpeta src, que contiene el documento mioCidFinal.docx y vamos a abrir la línea de comandos ahí. En Windows, sencillamente hay que abrir la carpeta correspondiente y hacer click en Archivo > abrir Windows Power Shell:



linea de comandos

y cuando se abra el Power Shell, vamos a escribir:

pandoc mioCidFinal.docx -f docx+empty\_paragraphs -t markdown --wrap=none --atx -o mioCidCampeador.md

Si ahora miramos de nuevo en la carpeta, veremos que un nuevo archivo (mioCidCampeador.md) ha aparecido en ella.

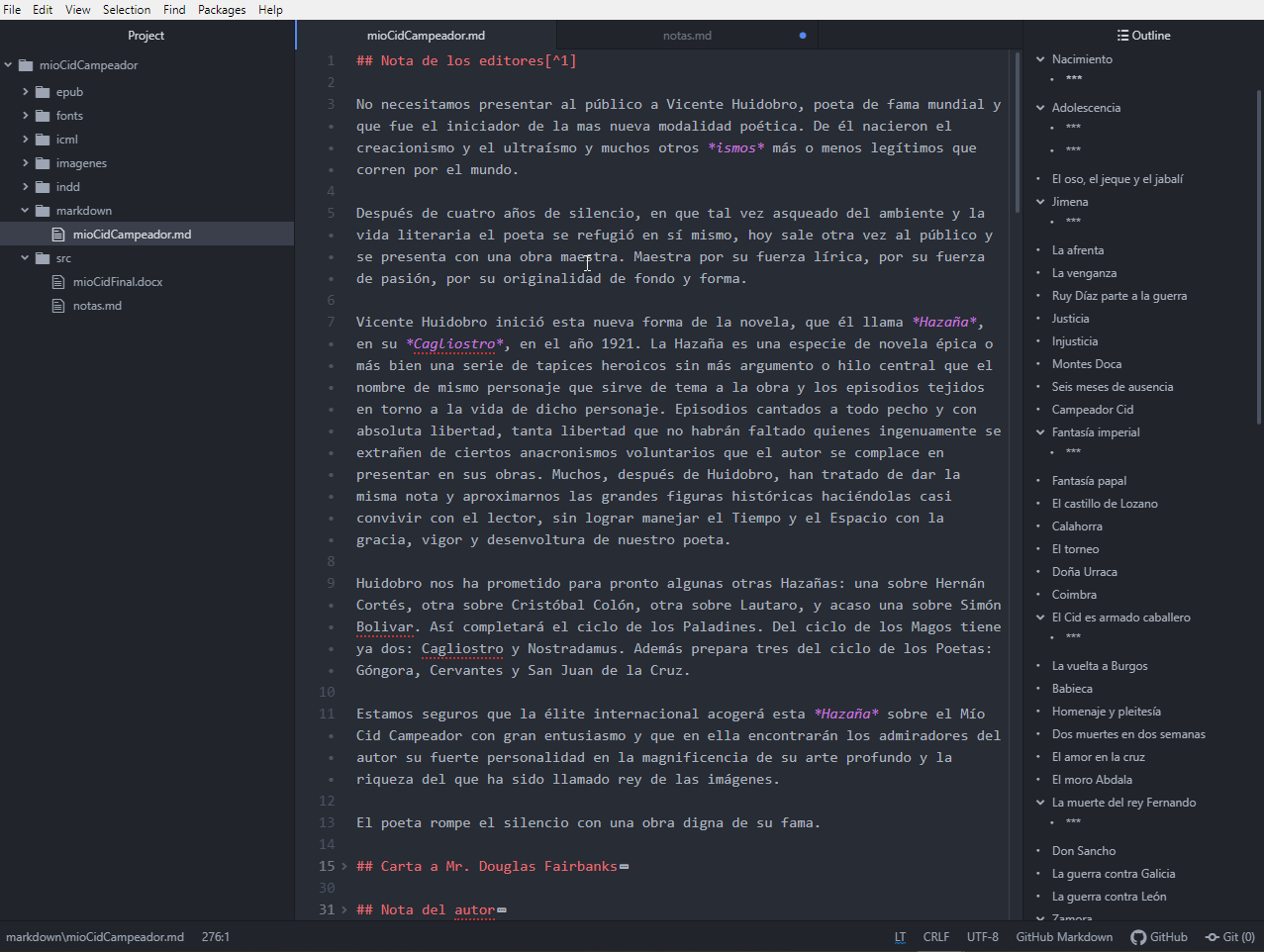
¿Qué hemos hecho? Paso por paso, es lo siguiente:

* hemos invocado el programa que vamos a utilizar (pandoc)
* le hemos dicho a qué archivo tiene que aplicar las opciones que vamos a definir (mioCidFinal.docx)
* le hemos dicho desde qué formato tiene que partir y cómo tratarlo: -f, de *from*, desde, en inglés; docx, es el formato de partida; la opción empty\_paragraphs le dice a Pandoc que queremos que limpie el archivo de todas las líneas vacías. Pandoc va a preservar todos los estilos predeterminados (encabezados) y estilos locales (itálicas y negritas), traduciéndolos a etiquetas en markdown, pero va a ignorar todos los estilos personalizados que hubiéramos podido definir en el documento (en caso de que quisieramos conservar los estilos personalizados de word, hay una opción: +styles).
* le hemos dicho a qué formato tiene que convertir el documento: -t markdown, a markdown.
* Le hemos dicho que elimine el ajuste de línea (word wrap) o saltos de línea no semánticos: wrap=none
* Que uniforme los encabezados (markdown tiene dos maneras de etiquetar encabezados, atx o setext-style headers): --atx
* y que con toda esta información, cree un archivo llamado mioCidCampeador.md

Pandoc ha hecho todo eso en un par de segundos.

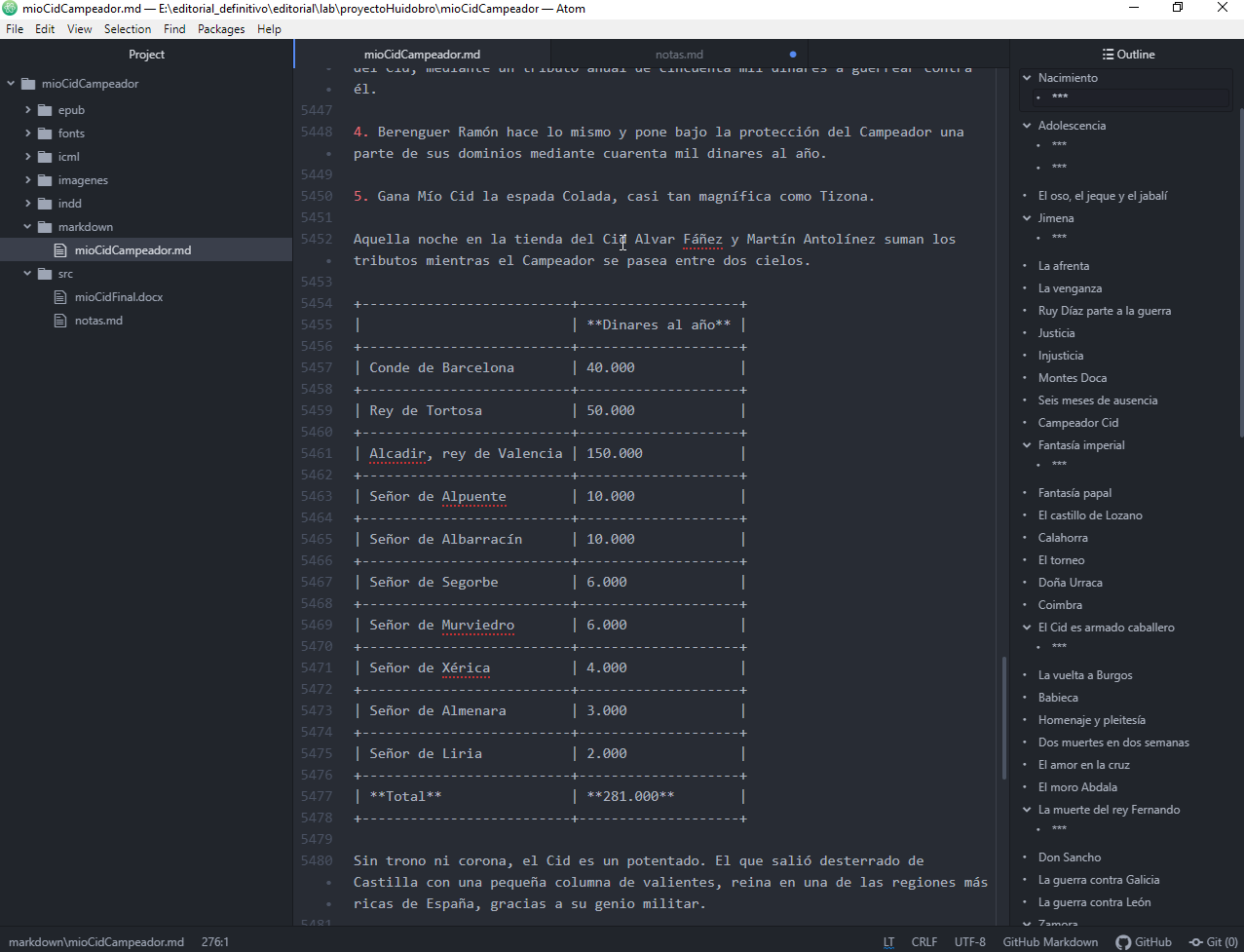
### Markdown

Ahora podemos abrir el directorio de nuestro proyecto con nuestro editor de texto y veremos el árbol de carpetas:



projectFolderEnAtom

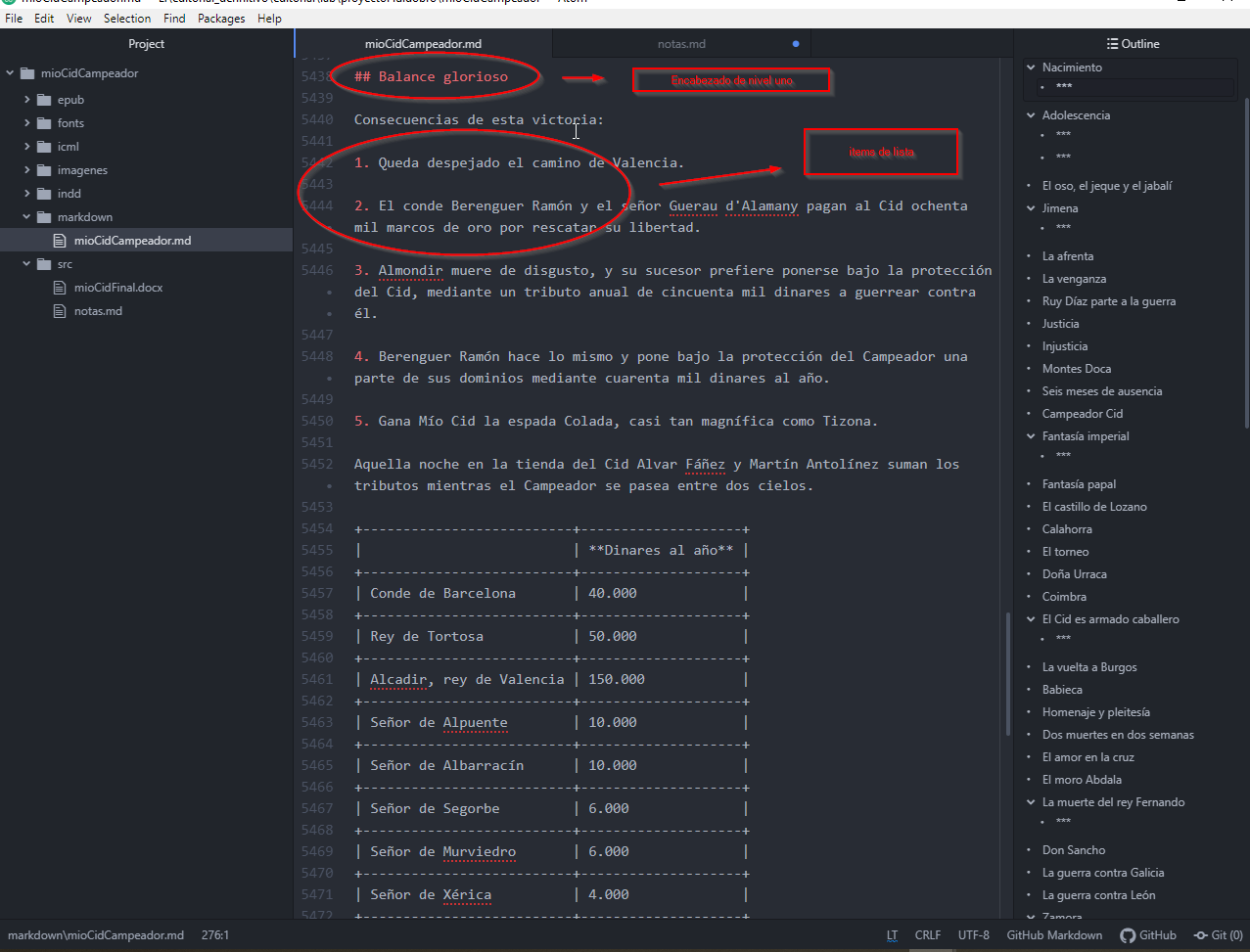
Abrimos el archivo mioCidCampeador.md (doble click en el panel de la derecha) y podemos ver lo que pandoc ha hecho. Los encabezados están correctamente distribuidos a lo largo del documento, el formato local se ha preservado. Incluso, podemos ver que Pandoc ha interpretado una tabla en el documento Word y la ha trasladado a markdown correctamente:



tablas (convertidas desde Word) en markdown

Como en cualquier manuscrito, es probable que halla numerosas correcciones que hacer, pero aquí tenemos una buena base. Volveremos al tema de las correcciones más adelante

De momento, vamos a sumir que la conversión fue exitosa. Podemos abrir el Outline del documento y verificar la estructura del documento, eso es ahora lo que más nos importa. Markdown tiene una sintaxis extremadamente sencilla de aprender. Si miramos el documento, podemos ver que: - los encabezados están marcados por hashes (#): # este es un título de primer nivel, ## este es un título de segundo nivel, etc… - las itálicas corresponden a las palabras (o frases) encerradas entre dos asteriscos (*palabra\_italicas*) y que las negritas lo están entre dos asteriscos (**esta\_es\_una\_palabra\_en\_negritas**). - Las tablas se construyen con una combinación de guiones y barras verticales. - Las listas numeradas, con el número, un punto y un espacio (1. item\_de\_lista) y las listas sin numerar con guión y espacio (- item\_de\_lista\_sin\_numerar)



ejemplo sintaxis markdown

Para una referencia de la sintaxis markdown, [aquí](http://https://pandoc.org/MANUAL.html#pandocs-markdown) está la página de referencia.

### Generar HTML a partir de Markdown

En un archivo HTML o EPUB (que en términos muy bastos no es otra cosa que un conjunto de archivos HTML: algo así como una página web comprimida y preparada para que la lea un dispositivo especializado) podemos distinguir dos secciones diferenciadas: el elemento head y el elemento body. El body es el contenido que es renderizado por un navegador (o por un software de lectura, en el caso de un archivo ‘epub’) y se muestra en la pantalla del dispositivo. El elemento ‘head’ en cambio contiene todo lo que necesita saber el software que va a renderizar la página para mostrarla correctamente:

<!DOCTYPE html>  
<html lang="en" dir="ltr">  
 <head>  
  
 <meta charset="utf-8">  
 <title>mi libro</title>  
  
 </head>  
 <body>  
 <h1> Este es el título del documento</h1>  
 <p> y aqui viene el cuerpo de texto</p>  
 </body>  
</html>

Este ejemplo muestra cómo se estructura un archivo ‘html’ muy básico. Pandoc es capaz de convertir un archivo markdown a HTML, incluyendo en éste toda la estructura semántica que hemos añadido al texto. Si convertimos el archivo del *Mio Cid Campeador* a HTML utilizando Pandoc (lo podemos hacer situándonos con el Power Shell en la carpeta donde está el archivo markdown y escribiendo: pandoc MioCidCampeador.md -f markdown -t html -o MioCidCampeador.html), va a convertir el archivo de esta manera:

<h2 id="nota-de-los-editores1">Nota de los editores<a href="#fn1" class="footnote-ref" id="fnref1" role="doc-noteref"><sup>1</sup></a></h2>  
<p>No necesitamos presentar al público a Vicente Huidobro, poeta de fama mundial y que fue el iniciador de la mas nueva modalidad poética. De él nacieron el creacionismo y el ultraísmo y muchos otros <em>ismos</em> más o menos legítimos que corren por el mundo.</p>  
<p>Después de cuatro años de silencio, en que tal vez asqueado del ambiente y la vida literaria el poeta se refugió en sí mismo, hoy sale otra vez al público y se presenta con una obra maestra. Maestra por su fuerza lírica, por su fuerza de pasión, por su originalidad de fondo y forma.</p>

¿Donde está el elemnto head? Lo que ha hecho Pandoc es convertir a etiquetas HTML todo el documento markdown, pero dado que no le hemos dicho explícitamente que queríamos un documento independiente, con sus elementos head y body, no lo ha hecho. Esto puede parecer una limitación, pero en efecto es extremadamente útil cuando lo que queremos es sencillamente HTML puro para insertarlo en otro documento. Para que haga el trabajo completo, tenemos que agregar la opción -s (por standalone, lo que a pandoc le dice que lo queremos es un documento independiente).

Si ahora reescribimos la orden: pandoc MioCidCampeador.md -f markdown -s -t html -o MioCidCampeador.html, van a pasar dos cosas: 1. Pandoc va a crear el documento de la manera que esperamos:

<!DOCTYPE html>  
<html xmlns="http://www.w3.org/1999/xhtml" lang="" xml:lang="">  
<head>  
 <meta charset="utf-8" />  
 <meta name="generator" content="pandoc" />  
 <meta name="viewport" content="width=device-width, initial-scale=1.0, user-scalable=yes" />  
 <title>MioCidCampeador</title>  
 <style>  
 code{white-space: pre-wrap;}  
 span.smallcaps{font-variant: small-caps;}  
 span.underline{text-decoration: underline;}  
 div.column{display: inline-block; vertical-align: top; width: 50%;}  
 div.hanging-indent{margin-left: 1.5em; text-indent: -1.5em;}  
 ul.task-list{list-style: none;}  
 </style>  
 <!--[if lt IE 9]>  
 <script src="//cdnjs.cloudflare.com/ajax/libs/html5shiv/3.7.3/html5shiv-printshiv.min.js"></script>  
 <![endif]-->  
</head>  
<body>  
<h2 id="nota-de-los-editores1">Nota de los editores<a href="#fn1" class="footnote-ref" id="fnref1" role="doc-noteref"><sup>1</sup></a></h2>  
<p>No necesitamos presentar al público a Vicente Huidobro, poeta de fama mundial y que fue el iniciador de la mas nueva modalidad poética. De él nacieron el creacionismo y el ultraísmo y muchos otros <em>ismos</em> más o menos legítimos que corren por el mundo.</p>  
<p>Después de cuatro años de silencio, en que tal vez asqueado del ambiente y la vida literaria el poeta se refugió en sí mismo, hoy sale otra vez al público y se presenta con una obra maestra. Maestra por su fuerza lírica, por su fuerza de pasión, por su originalidad de fondo y forma.</p>

1. Y nos va a arrojar un mensaje de error:

[WARNING] This document format requires a nonempty <title> element.  
Defaulting to 'MioCidCampeador' as the title.  
To specify a title, use 'title' in metadata or --metadata title="...".

Este mensaje significa que Pandoc ha sido capaz de crear el documento, pero dado que no le hemos indicado qué metadata utilizar, lo ha construido a partir de sus valores por defecto. Como, según la especificación, lo único imprescindible es que un documento tenga un título, Pandoc ha creado un título a partir del nombre del archivo.

### Metadata

Ya sabemos como Pandoc puede convertir automáticamente un documento markdown a HTML (sobre cómo se convierte a EPUB, lo veremos más adelante, pero el principio es el mismo). Lo que necesitamos ahora es aprender a alimentar a Pandoc con metadata para que este pueda escribirla en HTML (y en epub).

Existen varias maneras de preparar un archivo de metadata, la manera más simple es incluirla dentro del archivo markdown utilizando un yaml block.

[Yaml](https://en.wikipedia.org/wiki/YAML) es un lenguaje de serialización de datos, diseñado para ser legible tanto por software como por humanos y es el lenguaje que utiliza por defecto Pandoc para manipular metadatos. Se puede utilizar dentro de un archivo markdown pero también como un archivo independiente. En este último caso, el archivo tiene la extensión .yaml (matadata.yaml, por ejemplo). Para nuestro caso, lo integraremos dentro de nuestro archivo markdown. Un yaml block se escribe al principio del archivo markdown, utilizando tres guiones para encapsularlo:

---  
title: mi libro  
author: yo mismo  
date: 07-28-2020  
---

Este sería el ejemplo más elemental. A partir de esta información, pandoc puede poblar las etiquetas title y author en el header y también podrá utilizar esta información para crear un bloque de título en HTML con el título del documento, el autor, y la fecha en que fue creado. ¿Por que es bueno que hagamos esto? Porque la información que incluyamos en la cabecera del documento HTML una vez sea accesible en la web, es la información que utilizará el motor de búsqueda para indexar el documento. Una página web sin metadata es una página web invisible.

De la misma manera, Pandoc utiliza la metadata que especifiquemos para incorporarla en el archivo epub, de manera que luego esté disponible para que el dispositivo pueda mostrárnosla (en la biblioteca del aparato, por ejemplo).

En síntesis, Yaml es una manera de mapear etiquetas o variables con valores.

title (etiqueta) : Mío Cid Campeador (valor asignado a esa etiqueta)

Hay que notar que no hace falta utilizar comillas. Pero si el valor que estamos asignando contiene comillas (o cualquier caracter especial) es mejor entrecomillarlo (y utilizar las comillas simples para el texto que estamos asignando):

title: "Mío Cid Campeador, una 'relectura' de la historia del Cid"

También es importante tener en cuenta que Yaml es una manera de estructurar datos, no un lenguaje reservado o una especificación para expresar esos datos. Lo que significa que en realidad podemos utilizar las etiquetas o variables que necesitemos de manera arbitraria. Pandoc puede interpretar un conjunto importante de etiquetas, pero si nos parece insuficiente, podemos editar las plantillas y utilizar las que necesitemos (de esto hablaremos más adelante)

Ahora volvamos a nuestro libro. En la misma carpeta donde tenemos el archivo mioCidCampeador.md vamos a crear un nuevo archivo y le vamos a llamar metadata.yaml. En el vamos a escribir lo siguiente:

---  
title: Mío Cid Campeador  
subtitle: Hazaña  
author: Vicente Huidobro  
creator:  
- role: editor  
 text: Lucho Tapia  
  
identifier:  
- scheme: ISBN-13  
 text: 978-84-939173-9-5  
  
date: 2020-04-07  
  
description: "Publicada en Madrid en 1931, esta novela de Huidobro es una réplica del clásico Cantar De Mío Cid, en la que propone una reinvención del mítico héroe medieval español. A partir de una conversación con el actor norteamericano Douglas Fairbanks, quien lo entusiasma con la figura del Cid y le pide una recopilación de datos, el poeta se documenta abundantemente y descubre, en su propia genealogía, un lejano vínculo con el legendario Rodrigo. La fascinación crece y Huidobro, a su manera se reencarna en él, se viste con su armadura y se lanza en esta epopeya por los campos de batalla de la creación, derribando, una vez más, las rigideces expresivas contra las que siempre luchó y logrando una narración de vanguardia. Huidobro legitima su relato, tal como él mismo lo señala en el prefacio, como 'la verdadera historia del Mío Cid Campeador, escrita por el último de sus descendientes'"  
  
publisher: La Vorágine  
keywords: literatura chilena, surrealismo, creacionismo, literatura contemporanea  
rights: © 2020 La vorágine, CC BY-NC  
lang: es-ES  
cover-image: ../imagenes/el\_mio\_cid\_campeador.jpg  
css: ../epub/style.css  
toc: true  
---

Lo guardamos y luego volvemos al Power Shell y escribimos:

pandoc MioCidCampeador.md metadata.yaml -f markdown -s -t html -o MioCidCampeador.html

y lo que obtendremos será esto:

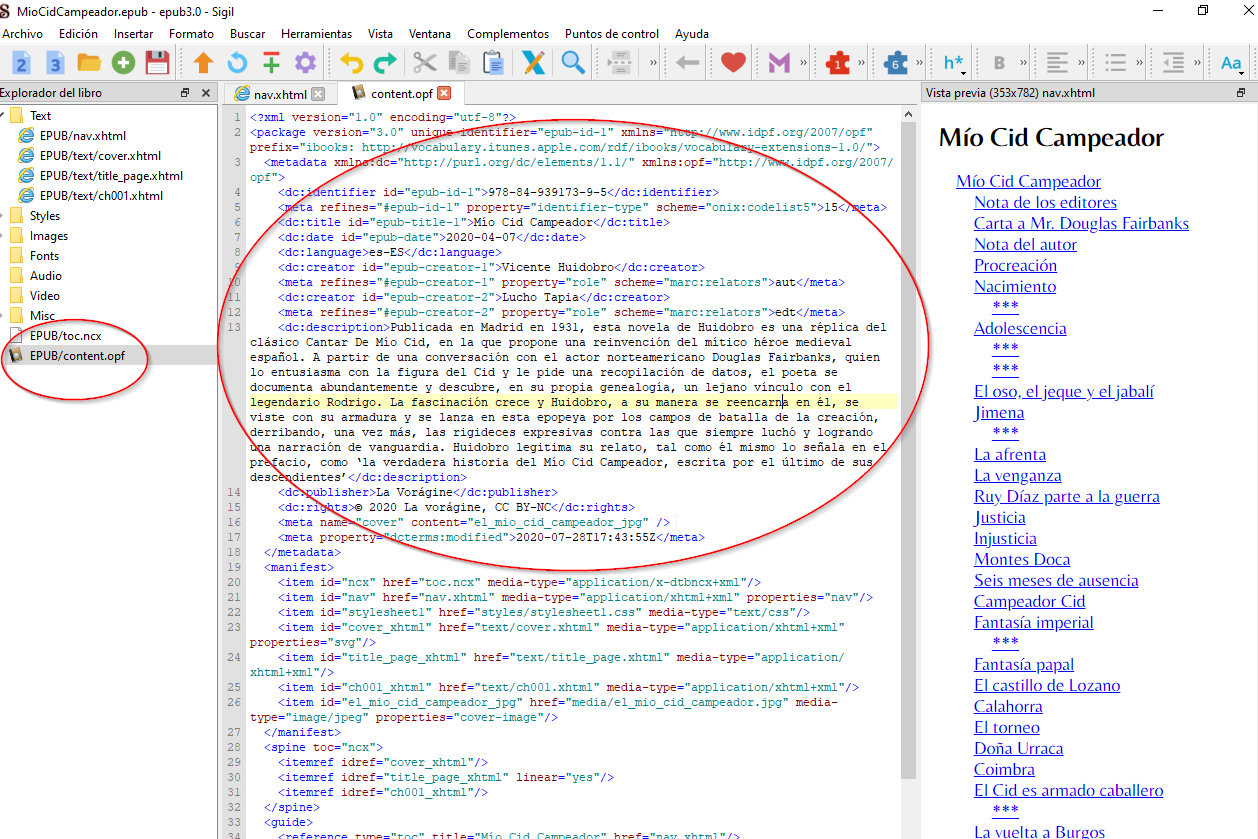


metadata incluida en html via yaml file

Nótese que también podemos generar un archivo epub:

pandoc MioCidCampeador.md metadata.yaml -f markdown -s -t epub3 -o MioCidCampeador.epub

y el resultado (si abrimos el epub y miramos en el archivo content.opf ):



epub metadata via yaml

En primer caso (conversión a html) pandoc creaba etiquetas <meta name="etiqueta" content="valor\_de\_la\_etiqueta" /> (<meta name="author" content="Vicente Huidobro" />), que es la manera standar para incluir metadata en html para la web. En el segundo caso, Pandoc introdujo la metadata utilizando los cuatro vocabularios reservados para metadata en la especificación epub3 (dcterms, marc relators, media y onix). Todo partiendo de un mismo archivo.

Más adelante veremos otros usos de Yaml y volveremos sobre la generación de Epubs.