Documentatón

Técnicas y herramientas ágiles y resilientes para escribir y publicar juntos

La comunidad de Grafoscopio

v 0.1.2 [ec74f6ba0b]

2019

Índice general

Preliminares

Presentación

Ilustración alusiva a la Torre del Reloj en Cartagena, Colombia

En pocas palabras

Este es un documento abierto, experimental y colaborativo sobre como crear documentos con esas tres propieades y por tanto se usa a sí mismo como ejemplo, pero se puede usar para obras similares. Está pensado para públicos diversos y proactivos, que no son expertos en tecnología pero tampoco se sienten intimidados por ella. Usa infrastructuras que encarnan los principios de agilidad y resiliencia y escarba en artefactos digitales contruidos comunitariamente para repensar el presente y el futuro.

Debería poderse usar en procesos de aprendizaje entre pares, sin mayor lectura o preparación previa por parte de un facilitador, sino presuponiendo que los aprendices leeran en conjunto y usarán la obra como guía de sus actividades prácticas, así como los canales comunitarios para lanzar preguntas sobre asuntos que no queden tan claros y visitarán futuras versiones de este manual, que pueden conseguirse en línea en:

https://mutabit.com/repos.fossil/documentaton/

De esta obra

Este es un texto extraño en sus pretensiones, métodos y lugares. Pretende ser un librito colectivo que se enseña y escribe en conjunto, desde un hackerspace en el Sur Global, llamado HackBo y se usa a sí mismo como ejemplo. Es decir, que acá verás las técnicas, metodos y herramientas que permitieron la escritura específica de esta obra, pero que pueden ser aplicados a ejercicios similares de escritura colectiva. Muestra el caracter abierto y procesual de un libro como cobra colectiva, en lugar de su concepción de producto relativamente cerrado, terminado e individual, que usualmente circula. Pues aunque todos sabemos que los libros una vez se leen quedan abiertos en muchos sentidos, este caracter conversacional y abierto casi nunca se resalta durante los procesos de escritura. Nuestra intención con este texto es precisamente explorar qué pasa cuando los procesos de lectura y escritura se abren mientras el libro se crea y no cuando está «terminado». En ese sentido, la obra en frente a ti es una forma de «inversión infraestructural», como diría Susan L. Star que busca poner al frente aquello que está en el fondo (infraestructuras, formatos, herramientas) para repensar aquello que está al frente (la escritura y los libros). Veremos más al respecto en la sección sobre los métodos, de esta misma presentación.

Esta obra regoje el trabajo de más de 400 horas de talleres de 6 días, llamados Data Weeks y sus contrapartes más breves, llamadas Data Rodas (de una tarde), que iniciamos a comienzos del año 2015 y han continuado hasta el momento. En ellos indagamos colectivamente por cómo cambiamos los artefactos digitales que nos cambian (esta fue la pregunta doctoral de uno de los promotores de esos encuentros) y por el

camino, juntamos diversidades, celebramos, bebimos y recorrimos caminos adjacentes. En particular los referidos a otras maneras y artefactos para escribir, pues pensamos que a los monopólicos lugares y herramientas comunes las falta mucho para pavimentar nuestra imaginación y nuestos hábitos frente a cómo y donde escribir. Aún podemos concebir, resistir e imaginar desde otras formas de escribir, con ventajas y perspectivas particulares, como este texto muestra.

También creemos que estas técnicas y maneras de hacer reflejan un espíritu de la época donde está transición está ocurriendo y muestran al libro como proceso colectivo, en lugar de como producto individual. Esfuerzos como LeanPub, los book sprints, FLOSS Manuals, Unglue.it y GitBook, dan cuenta de esta manera de repensar los procesos escriturales que han estado instaurados entre nosotros desde hace centurias, de nuevas maneras, gracias a las tecnologías digitales disponibles. Nuestros procesos han sido desarrollados en paralelo desde hace décadas, pero con una perspectiva y anclaje en el Sur Global y en este momento confluyen, divergen y dialogan con los ejemplos antes mencionados. Este texto es una forma de mostrar un recorrido en particular que se instaura en esos lugares de frontera, poco conocidos, pero cada vez más visibles y potentes.

Notarás como esta obra evoluciona y pasa por distintos estados de madurez, heredando algunos vocabularios del mundo del software, juntando metodologías del performance, la hackatón (maratón de prototipado), y el taller, entre otros. Es también un artefacto amalgamado, que incluye un canal de mensajería, una lista de correo, repositorios de código y herramientas de software (descritas en detalle acá) y, por supuesto, el resultado en distintos formatos PDF, EPUB, código fuente (lo más seguro es que actualmente te encuentres leyendo alguno de ellos). Ahora se encuentra en estado alfa y pasará por un conjunto de estados beta sucesivos hasta que el proceso logre cierta homeostasis (equilibrio dinámico con el entorno), para lo cual tu participación en este proceso puede ser clave Así que puedes reencontrarte con la obra y la comunidad a su alrededor en sus diversas formas, haciendo click en los enlaces previos. Este texto explicará en detalle algunas formas de participación en dichos sitios.

De nuestros lectores: proactivos y diversos

Esta cartilla está pensada para cualquiera que quiera construir textos en formato digital con otras personas para expresar conocimientos colectivos. Ya seas un científico, académico, activista, periodista, hacker, creemos que esta manera de hacer colectivamente es potente y se instaura en la idea de escritura como performance estético colectivo en lugar de como forma de escritura individual.

Presuponemos un perfil de nuestros lectores: los imaginamos como diversos y proactivos. No son expertos en tecnologías digitales forzósamente, pero no se sienten intimidados con ella. Quieren explorar estas

nuevas formas de escribir y apropiarlas con práctica constante (como hemos hecho todos). Si te sientes identificado con esta manera de pensar la escritura y la colaboración, bienvenido a este texto. A lo largo del mismo te diremos como participar de él de distintas maneras, desde comentarios en línea a procesos de coescritura y repositorios de código.

Esperamos que, como todo libro, este sea el comienzo de una conversación.

De nuestros principios: agilidad y resiliencia

Lo digital se carateriza por su fragilidad: copias y versiones que se pierden, cambios sin control, ausencia de historia. Pero no tendría por qué ser así. Desde este texto apostamos por otra manera de pensar lo digital que tienen que ver con su *agilidad* (alta adaptabilidad dinámica) y su *resiliencia* (capacidad de recuperar a perturbaciones). Estas ideas son complemetarias en lugar de contradictorias. Si una técnica es ágil, puede cambiar y adaptarse rápidamente al contexto. Y si es resiliente, puede recuperar su forma a pesar de las perturbaciones externas. Es decir ambas propiedades son maneras de lidiar con el cambio, lo cual es un problema esencial de la informática y nuestra manera de lidiar con el cambio es esencialmente ofrecer robuztes y flexibilidad para adaptarse al cambio, pero también para soportarlo. En últimas, hablar de agilidad y resiliencia, es hablar de cambio e identidad: los documentos que generemos acá preservaran su identidad y daran cuenta de cómo cambio, además de que permitiran que múltiples personas y coautores ayuden tanto con los cambios como a preservar los documentos y su historia.

De nuestro método: meta arqueología digital activista

Como dijimos este texto se basa en más de 400 horas de talleres colectivos, una pregunta doctoral de varios años, y se suma a un proceso de comunidades de software libre y de código abierto en latino américa y el sur global de un par de décadas. Además invita a saberes desde otros lugares, subjetividades y comunidades. También mencionamos que hacemos una «inversión infrastructural» poniendo aquello que está en en fondo al frente y viceversa (término acuñado por Susan L. Star). Así, al revisar las metodologías e infraestructuras que nos permiten escribir, podemos repensar la escritura misma y sus resultados (obras, libros, artículos, caritllas, novelas, etc).

Hemos acuñado un término para esta forma de reflexión sobre los artefactos y la práctica propia desde la diversidad y el pasado para entender el presente y reconfigurar el futuro, que dialoga con las propuesta de

Star y nuestras propias realidades e intereses. Le llamamos **meta arqueología digital activista**. Veamos por qué:

- **Arqueología:** explorar los artefactos del pasado para entender el presente y potenciar el futuro.
- **Digital:** Usa permanentemente herramientas y métodos digitales, sin limitarse a ellos.
- **Activista:** Se piensa en clave de derechos (queremos alentarlos con estas prácticas) y la idea de co-diseñar y co-construir un mundo compartido.
- **Meta:** Usar las técnicas digitales, no sólo para entender a otros, sino a nosotros, es decir es una práctica que también repiensa y rehace lo que hacemos, es decir, es auto-transformativa. Para esto usamos el caracter auto-referencial de lo digital, es decir que cuando tenemos acceso al código fuente (las instrucciones de cómo opera un sistema) podemos cambiarlo.

Esta es una definición en construcción, como la obra y los espacios en los cuales se inscribe y te damos la bienvenida para construirla y criticarla en conjunto.

De nuestras herramientas: infraestructuras de bolsillo

Queremos que las infraestructuras, que son posibilitadores de acciones y relacionales, sean plurales y para ello hemos elaborado el concepto de *Infraestructuras de Bolsillo*, las cuales son sencillas, auto-contenidas (es decir no requieren de muchas dependencias externas) y funcionan bien en y fuera de línea.

Como *infraestructuras de bosillo* denominamos a aquellas infraestructuras que son:

- Simples.
- Autocontenidas.
- Funcionan bien en y fuera de línea.
- Se ejecutan en un amplio rango de hardware y software, desde computadores modestos hasta servidores potentes, sobre distintos sistemas operativos.

Dichas infraestructuras van contra lo que llamamos el *Feudalismo de metadatados*, que se refiere al hecho de que ciertas tecnologías y redes actuales favorecen que haya copias de la información en distintos dispositivos, pero los metadatos, que son producto de la interacción y labor social, sólo quedan en infraestructuras ajenas. Así, nuestras fotos están en nuestros celulares y en las redes sociales y nuestro código está en nuestros computadores y en los repositorios de código, pero los *favs*, los comentarios, hechos tanto a las fotos como al código, se quedan sólo en esos servidores centrales (Twitter, Facebook, GitHub), apropiándose del valor social que se creó de manera social y en comunidad.

Ejemplos de Infraestructuras de bolsillo

- PortableApps.
- Grafoscopio.
- Fossil.
- Red Language.
- ULua: Universal Lua Distribution

Referencias Extra:

- Should We Treat Data as Labor? Moving Beyond «Free».

Convenciones de lectura

Imagen alusiva a la lectura

Tipográficas e iconográficas

Este conjunto de convenciones te ayudarán mientras estás leyendo este documento:

- La *cursiva* y la **negrilla** son usados para énfasis y énfasis fuerte, respectivamente
- La citas son presentadas de esta manera:

> Mantente alerta de las noticias y citas falsas en Internet.
>
> –Benjamin Franklin, 1841.

- Las fuentes monoespaciadas son usadas para comandos y ubicaciones en el computador, como: copia el archivo `algo.txt` a la carpeta documentos.
- Las jerarquías de menús gráficos de programas están organizados usando el separador >. Por ejemplo, decir «ve a `Notebook > open...`», significa que el lector debería hacer click en el menú `Notebook` y luego buscar el submenú `open...`

Tenemos actividades que son más informativas y te darán una panorámica sobre un tema particular, mientras que otras son más prácticas. Usamos íconos para distinguirlas:

- : Lectura.
- (tiempo): Ver un video. El número en paréntesis es la duración.
- : Actividad práctica.

También usamos varios avisos para resaltar piezas de informacion:

:point_right: Usado para mostrar información resumida de un tema particular.

:heavy_exclamation_mark: Usado para mostrar alguna información de la cual necesita estar al tanto.

Versiones y revisiones

Como hemos dicho, este documento es un objeto mutable y móvil. Para acercarse al mismo hemos usado lo que se conoce como versionado semántico, en el que un número corresponde identifica de manera únivoca la copia del documento que se está leyendo, siguiendo las siguientes convenciones:

- La versión de la obra tiene a lo sumo tres números de la forma `x.y.z`:

    - `x` es la versión *mayor*, que habla de cambios grandes en el texto que son incompatibles con la documentación del pasado, por ejemplo, si varias de las técnicas e infraestructuras explicadas ha sido cambiada sustancialmente.
    - `y` es la versión *menor*, cuando se adiciona funcionalidad de manera compatible con la documentación previa, por ejemplo nuevos (sub)capítulos que explican técnicas e infraestructuras nuevas o amplían las existentes.
    - `z` es la versión de *remiendo*, que se refiere a correciones que son compatibles con las versiones previas, por ejemplo las ortográficas.

- La revisión, que es una combinación de números y letras (o alfanumérica), referida a un identificador único de los archivos fuentes a partir del cuales se generó la obra derivada (en PDF, HTML o EPUB). Dichos archivos fuentes están listado en un archivo `manifest.ston` dentro de la carpeta de cada idioma que la obra tiene.

La copia en tu poder está definida así, de forma unívoca, por dos items: la versión y la revisión que aparecen en la página de portada, acompañados por otros datos como los autores, el título y la licencia de la obra. La versión y la revisión también aparecen en la esquina inferior izquierda de varias páginas impares del texto (salvo en aquellas que son, además, inicio de capítulo). Ten presente tanto el número de versión como el de revisión cuando uses nuestro sistema de reporte de incidencias, como explicamos en la sección titulada «Contacto y Ayuda».

Contacto y Ayuda

Si necesitas recibir o brindar ayuda respecto al contenido de esta obra, tenemos varias formas de contacto:

- Para la comunidad de Grafoscopio:
    - El canal de Telegram.
    - La lista de correo:
        - Archivos históricos.
        - Suscripción.
- Para comentarnos sobre tus lecturas y sugerencias, usa la página de hyphotesis, revisar todos los *tickets* para participar de las conversaciones allí ó crea tu propio *ticket*. Cuando crees un nuevo *ticket*, recuerda mencionar la versión y revisión de la obra a la que haces referencia, siguiendo nuestras conveciones de versionado semántico. El capítulo de Fossil en esta misma obra incluye información más detallada sobre el uso de *tickets*.

Si tu papel es como lector o editor varias preguntas orientadoras pueden guiar tu acción:

- ¿Está este texto claro?, ¿está bien redactado?
- Si yo fuese un lector novato, ¿lo entendería o sería más un texto acompañante de una práctica donde se explica de manera más completa lo que no está claro del todo?,
- ¿Esta traducido por completo o veo porciones en Spanglish?,
- ¿Las gráficas quedan bien escaladas cuando la exporto a distintos formatos?, ¿todas las gráficas tienen descripciones («captions»)?

El libro mismo explica en detalle los conocimientos necesarios para participar de la cocreación del mismo. Las reglas generales son sencillas:

- Todas las preguntas son bienvenidas, pero nos ayudas mucho a ayudarte si puedes ser muy detallado al respecto: si ya has buscado en Internet, qué respuestas obtuviste y cómo intentaste seguirlas; si

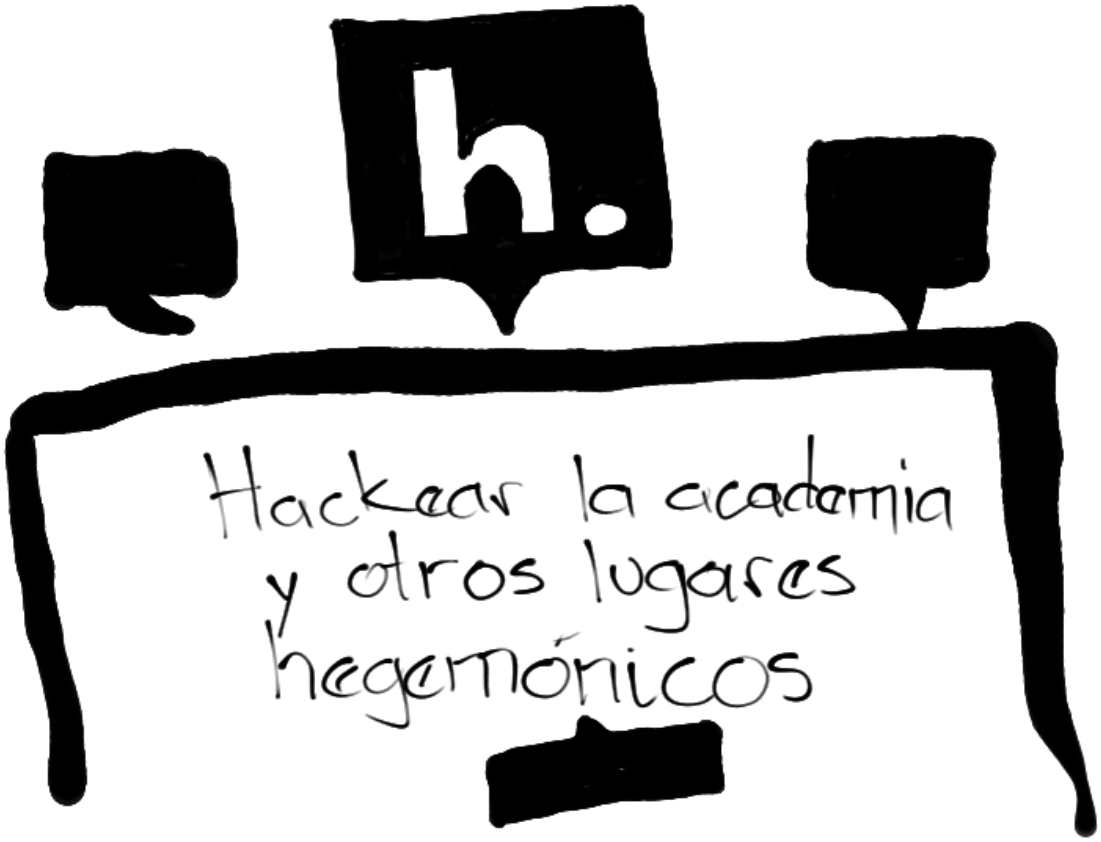
tuviste un mensaje de error, qué es lo que dice; si estás probando el software en qué contexto (con cuál sistema operativo y con cuáles versiones del software).

- Si vas a aportar a un capítulo ya existente, hazlo manteniendo el estilo editorial de la obra hasta el momento y consulta con los autores o por los medios de contacto previos, en caso de no tener claridades que necesites antes de empezar o tengas dudas al respecto. Puedes colaborar respondiendo a las preguntas de la parte previa, de modo que vayas complementando y mejorando traducciones, redacción, figuras y descripciones, entre otros.
- Si quieres escribir un capítulo agrégalo al repositorio y haz un bosquejo del mismo indicando sus subcapítulos y explica en una(s) frase(s) qué debería ir en cada una de ellos.

  Al final de uno o varios capítulos crea una lista de verificación de lo que consideras que un aprendiz debería estar en condiciones de hacer una vez ha pasado por ellos (mira el libro para ejemplos de tales listas de verificación). Estas listas pueden ser escritas incluso antes de tener mayores detalles sobre los contenidos, pues a partir de ellas podemos ver qué es necesario escribir para que los saberes que verifican puedan ser explicados y adquiridos.

Leer juntos

Hypothes.is: anotar y conversar en documentos en línea



Ímagen alusiva a la presentación del capítulo

Esta es una guía en español para usar la herramienta de anotaciones abiertas hypothes.is. En términos

generales Hypothesis permite comentar cualquier documento en web, añadiendo una capa adicional que permite subrayar, comentar, etiquetar o responder a comentarios de documentos web como sitios, pdfs, videos y otros.

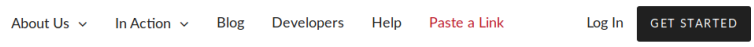
Al terminar este manual, usted estará en capacidad de:

- Crear una cuenta en Hypothes.is
- Acceder a su cuenta
- Incluir documentos en su cuenta para comentar.
- Subrayar textos.
- Crear y etiquetar comentarios.
- Responder a comentarios de otros.
- Compartir sus comentarios.

Crear una cuenta

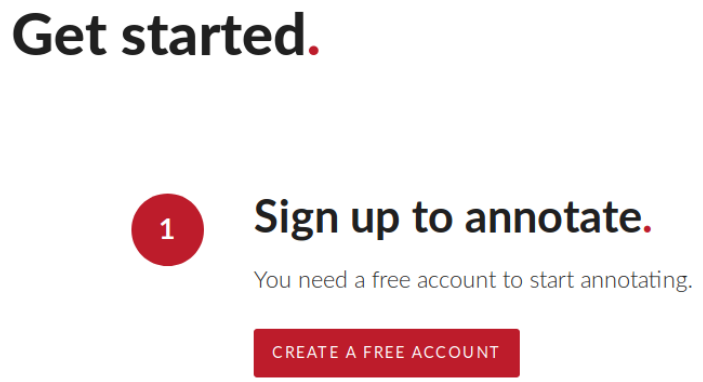
Si bien usted puede comentar documentos de manera anónima, al crear una cuenta usted tendrá un listado con los documentos que ha comentado en hipothes.is. Así se crea una cuenta:

- Acceda al sitio web https://www.hypothes.is/
- Seleccione el botón «*Get started*» como se ve en la imagen a continuación:

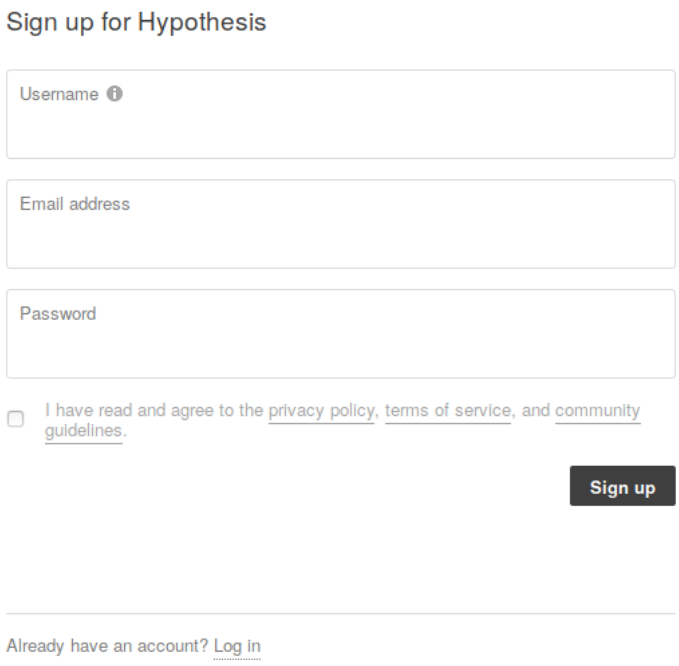


Creación de cuenta: acceda a Get started

- Acceda al botón «*Create a free account*» como se ve a continuación:



- Escribe tu nombre de usuario, dirección de correo, contraseña y acepta los términos y condiciones y selecciona el botón «*Sing up*», en la pantalla que se ve a continuación:



¡Listo! Acabas de crear tu cuenta. La próxima vez que uses hypothes.is simplemente deberá escribir su nombre de usuario y contraseña.

Acceder a tu cuenta

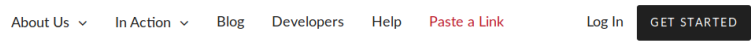
Si ya tienes una cuenta en Hypothesis hay dos formas de acceder a tu cuenta:

- Desde el sitio principal de Hypothesis
- Desde un documento que se esté comentando en hypothesis

Acceder desde el sitio principal de Hypothesis

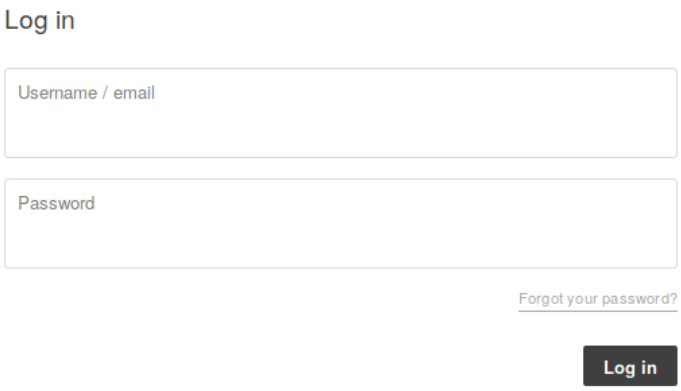
- Acceda al sitio web https://www.hypothes.is/
- Seleccione el botón «***Log in***» como se ve en la imagen a continuación:

Cuidado: El acceso o ***log in*** se encuentra al lado izquierdo del botón resaltado en negro.



Creación de cuenta: acceda a Get started

- Aparecerá un cuadro que solicita su nombre de usuario o correo y su contraseña como se ve en la imagen a continuación:

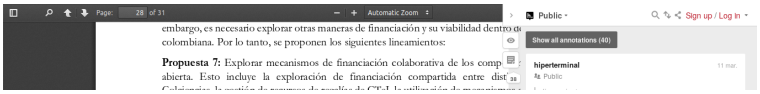


¡Listo! Después de diligenciar estos datos usted ya estará usando Hypothesis desde su cuenta.

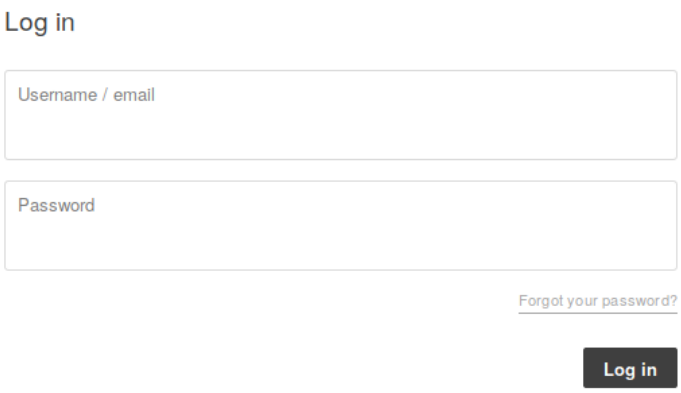
- Si olvidó su contraseña puede usar el enlace de " **Forgot your password?**"

Acceder desde un documento que se esté comentando en hypothesis

Reconocerá los documentos que pueden comentarse desde Hypothesis porque en el lado derecho de su pantalla encontrará una pestaña gris como la que se ve a continuación:

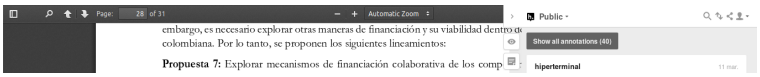


- En la columna de la izquierda seleccione la opción ***Log in***
- Aparecerá una ventana emergente con las opciones de usuario/correo y contraseña, como se ve a continuación:



¡Listo! Después de diligenciar estos datos usted ya estará usando Hypothesis desde su cuenta.

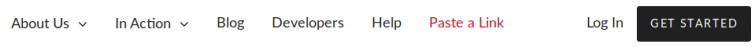
Reconocerá que está usando hypothesis desde su cuenta porque aparecerá la figura de una persona en el extremo superior derecho, como se ve a continuación:



Incluir documentos en su cuenta

Luego de acceder a su cuenta usted podrá incluir documentos en su listado personal para que puedan ser comentados. Para ello deberá:

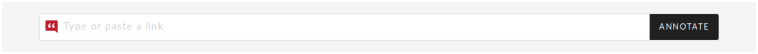
- Acceder al sitio web principal de Hypothesis, disponible en https://www.hypothes.is/
- Acceder a la opción «***Paste link***» como se ve a continuación:



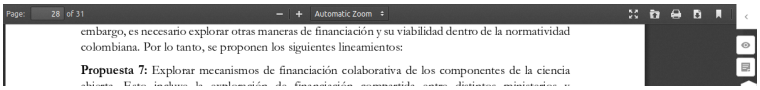
Agregar documentos para comentar: Paste link

Cuidado: La opción de pegar un enlace aparece en rojo en la imagen y no necesita acceder a su cuenta para que esté activa.

- Le aparecerá un cajón para incluir un enlace al documento que quiere anotar, seguido por el botón ***Annotate*** como se ve a continuación. Pegue el enlace del documento a anotar y seleccione el botón Annotate:



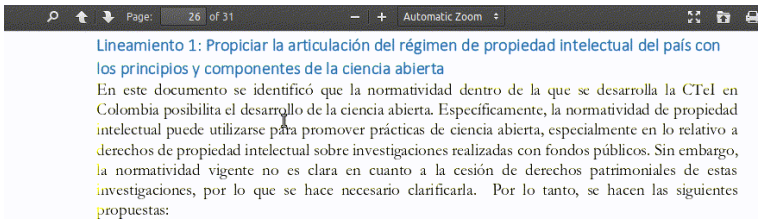
- Le aparecerán las pestañas de la derecha, con tres opciones como se ve a continuación. Seleccione la pestaña **<** para abrir la columna de la derecha, donde podrá ver las opciones de Hypothes.is



Una vez haya subrayado un texto o creado una anotación, el documento aparecerá en su historial de usuario.

Subrayar textos

Mientras está revisando un documento con hypothesis, puede subrayar una parte del texto y aparecerá la opción de *Anotar* (*Annotate*) o *Subrayar* (*Highlight*), como se ve en esta imagen. Haga click en *Highlight* y el documento incluirá un resaltado de color amarillo.

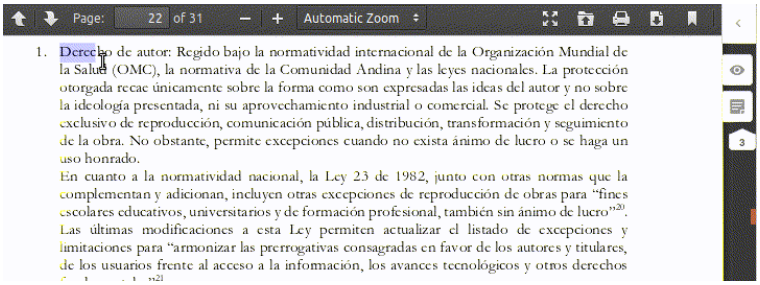


Subrayado en hypothes.is

¡Listo! Siguiendo estos pasos usted ya habrá subrayado un texto con Hypothes.is

Crear anotaciones y etiquetas

Mientras está revisando un documento con hypothesis, puede subrayar una parte del texto y aparecerá la opción de *Anotar* (*Annotate*) o *Subrayar* (*Highlight*), como se ve en esta imagen.



Haga click en *Annotate* y el documento incluirá un resaltado de color amarillo, adicionalmente en la sección lateral derecha se desplegará un cajón de comentarios, donde podrá escribir su comentario. Adicionalmente aparece un campo de etiquetas (en inglés *Add tags..*) donde podrá incluir palabras para describir su comentario, para que a futuro con esa palabra pueda buscarlo y encontrarlo.

Finalmente deberá presionar el botón *Post to public* o Publicar públicamente. Esto quiere decir que su comentario podrá ser visto por quien tenga acceso al documento en Hypothesis.

¡Listo! Siguiendo estos pasos usted ya habrá incluido un comentario a un texto con Hypothes.is

Responder a comentarios de otros

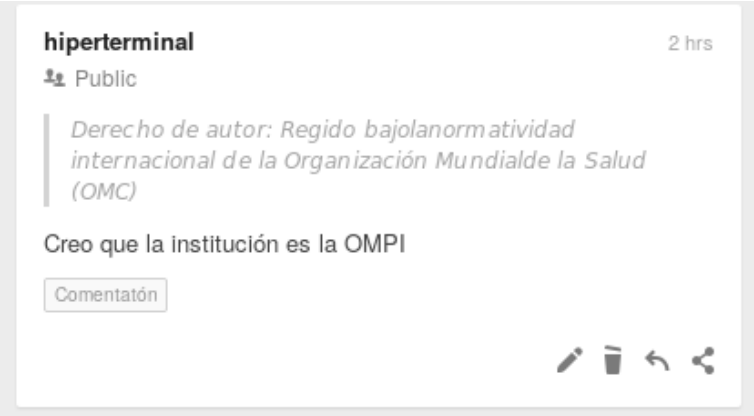
En la columna lateral derecha se encuentran las anotaciones y comentarios que se han hecho a un documento. Hypothesis permite crear hilos de debate en comentarios específicos. Esto se hace así:

Vaya hasta el comentario que quiera responder. En la parte inferior derecha encontrará una flecha hacia la izquierda, el tercero de cuatro íconos, que deberá pulsar.

Esto abrirá un cajón de texto debajo del comentario inicial, en el cual podrá escribir su comentario, como se ve a continuación. Podrá escribir las etiquetas que identifiquen a este nuevo comentario, para describirlo.

Una vez redactada su respuesta e incluidas las etiquetas deberá pulsar el botón *Post to Public*. Así su comentario será visible para otras personas que accedan al documento en Hypothesis.

Cada comentario puede tener múltiples respuestas anidadas. Éstas se enumerarán y se podrán desplegar o esconder haciendo click, en el caso de la siguiente imagen, en *Hide replies* que aquí identifica que el



Anotación en Hipothesis

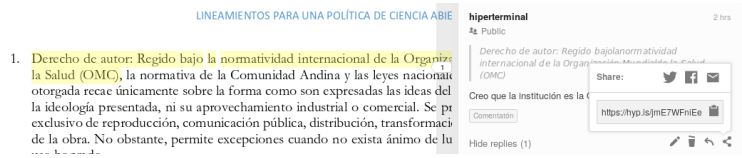
comentario inicial tiene una respuesta. Si usted recibe un comentario, Hypothesis le enviará un correo electrónico notificándolo de dicha acción.

¡Listo! Siguiendo estos pasos usted ya habrá respondido a un comentario en Hypothes.is

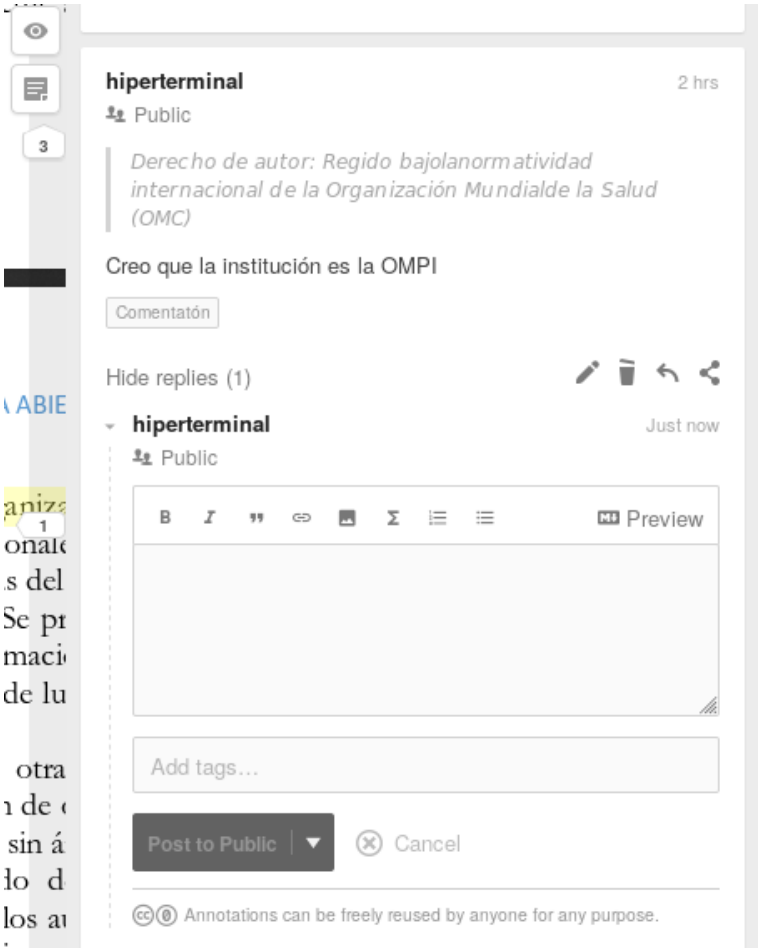
Compartir sus comentarios

Para compartir un comentario específico seleccione en el comentario, el cuarto ícono de los que se encuentran en la parte inferior derecha de la barra lateral derecha, como se muestra a continuación, el ícono con tres puntos unidos por una línea.

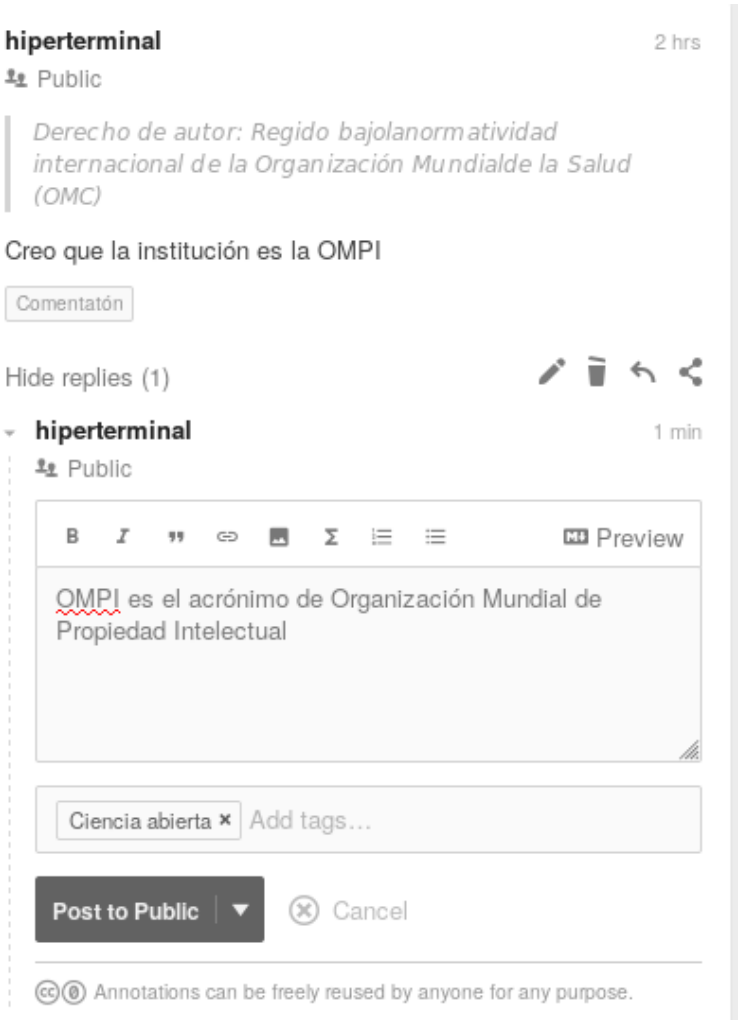
Se desplegará una ventana en la que podrá copiar la *URL* para compartir, o al lado del texto *Share* (*Compartir* en inglés) aparecen los íconos de Twitter, Facebook o un correo electrónico. Podrá enviar el enlace por cualquiera de estos medios de forma automática.



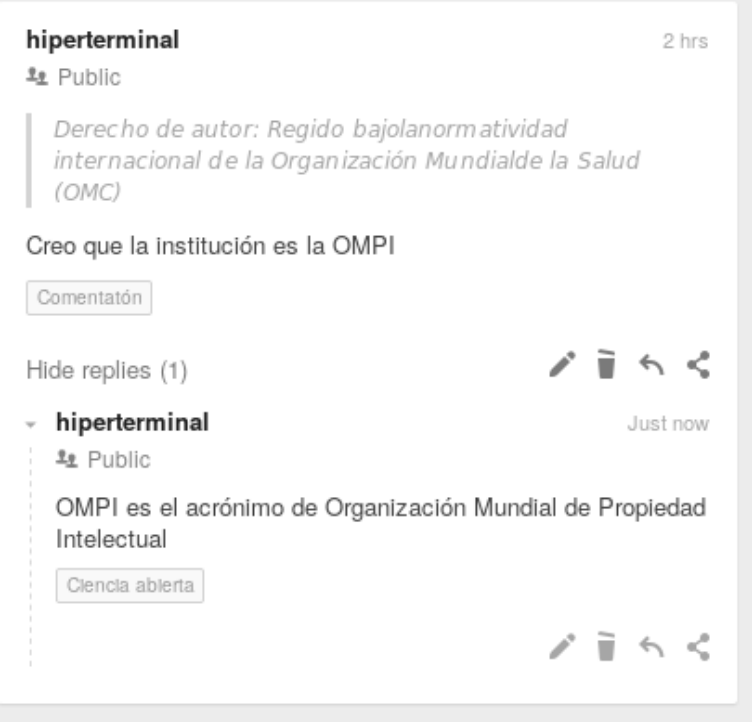
¡Listo! Siguiendo estos pasos usted podrá compartir un enlace específico a un comentario mediante Hypothes.is



Cajón para responder a un comentario en Hypothesis



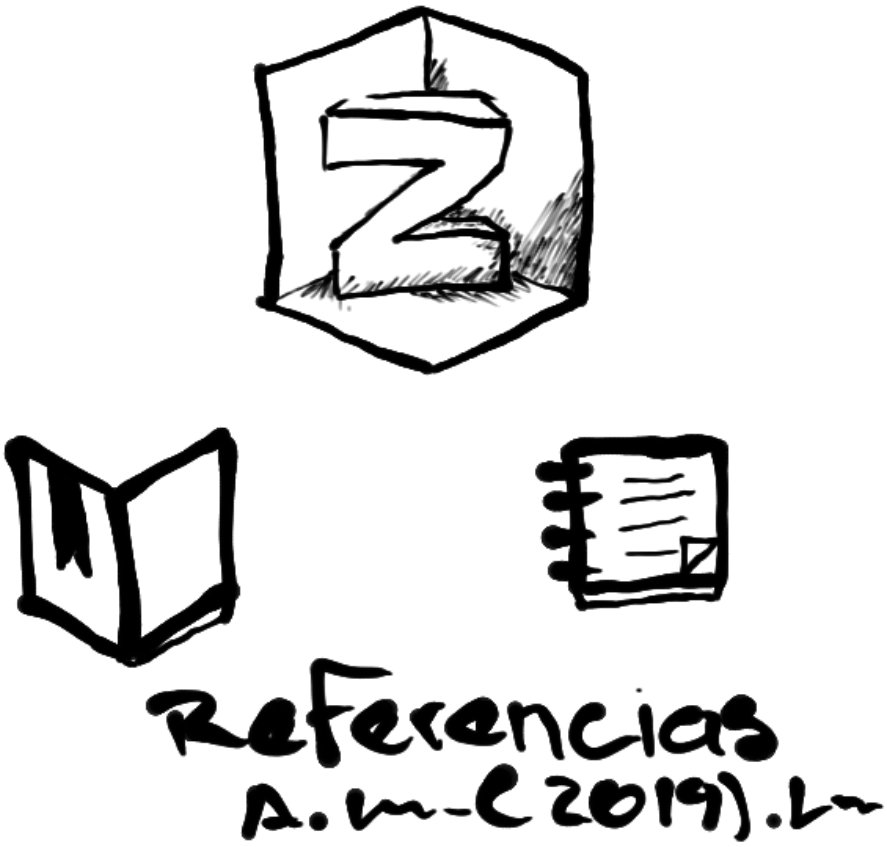
Cajón para responder a un comentario en Hypothesis



Respuesta a un comentario de Hipothesis

Anotación en Hipothesis

Zotero: Gestionando referencias bibliográficas



Un gestor de referencias bibliográfico te ayudará a organizar tus lecturas, tu biblioteca digital para mantener registro de lo que lees. Además puedes compartir bibliografías o crearlas colaborativamente. Con esta herramienta organizamos lo que leemos, incluso grupalmente.

Cuando escribe un trabajo, las referencias bibliográficas pueden ser un problema. ¿No ha tenido que enfrentarse a la revisión minuciosa de los diferentes estilos de citación para que su texto cumpla estándares

de trabajos escritos? Seguramente APA, Chicago o si está en Colombia ICONTEC son estilos de citación que podrían recordarle de lo que nos referimos. Zotero es una de nuestras infraestructura de bolsillo para automatizar este trabajo, hacerlo más ágil e incluso leer con otros.

Al terminar esta sección usted podrá:

- Gestionar sus referencias bibliográficas personales
- Crear o participar en grupos que comparten referencias bibliográficas

La instalación de Zotero

Hay bastante documentación sobre esto, así que en vez de repetirlo, puedes ir a este manual donde te explican su instalación y funcionamiento, o este otro documento que además es libre donde explican cómo instalar Zotero y sus funcionalidades básicas. Ya que puedes consultar estos recursos, que además son bastante populares entre las bibliotecas (y puedes ir a tu biblioteca más cercana a que te den una mano si la necesitas), nos concentraremos en cómo usamos desde la *Comunidad de Grafoscopio y Dataweet* esta herramienta.

La instalación resumida

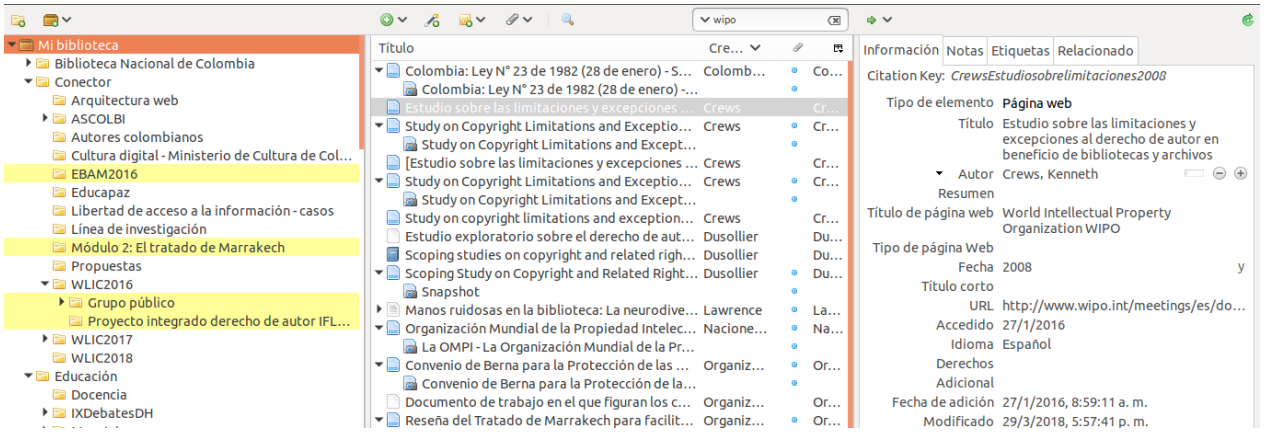
Es realmente simple:

- Ve al sitio de descargas de Zotero
- Descarga
- …y ejecuta.

No tienes que usar líneas de comando de la terminal de tu computador. Han hecho un buen trabajo en la usabilidad de los instaladores, así que no debería darte problemas.

Gestionar sus referencias bibliográficas personales

Puede crear Carpetas o subcarpetas para gestionar tus referencias personales. Creo que algo que no se ha dicho mucho en los diversos manuales y herramientas es que el espiritu de esta herramienta es evitar la duplicidad de trabajo. No necesitas crear varias veces la misma referencia si la vas a usar muchas veces en carpetas diferentes.



Una misma referencia (en la columna de la mitad) se está usando en cuatro diferentes carpetas, resaltados en amarillo en la columna de la izquierda

En la anterior imagen puede verse como una única referencia resaltada en el panel de la mitad, es usada por varias carpetas, resaltadas en el panel de la izquierda. Esto implica que si se actualiza una referencia con datos en el panel de la derecha, todos los folders que la usan se actualizarán con estos datos.

Referencias bibliográficas automáticas

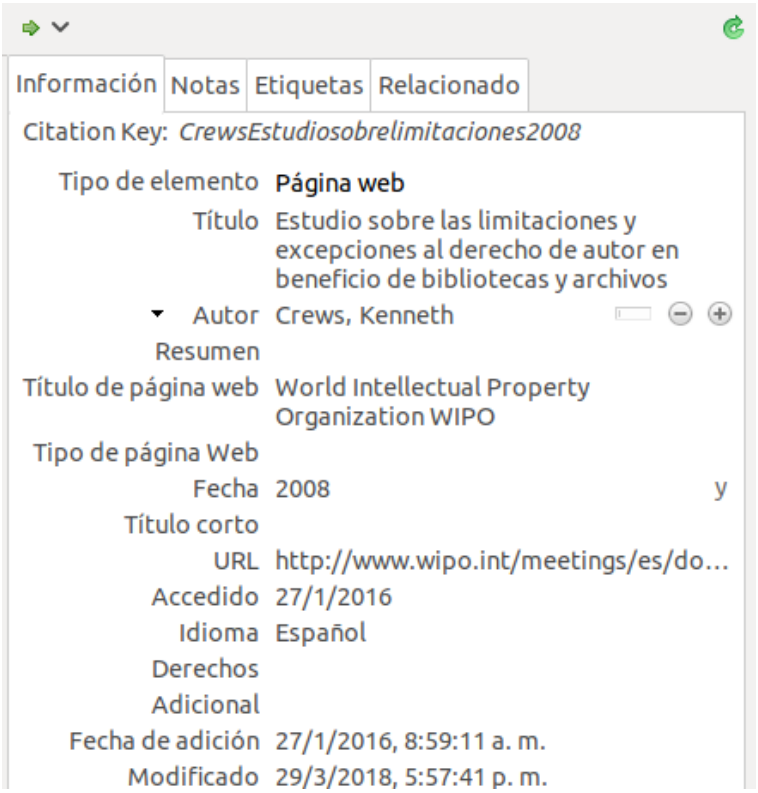
Quizá la utilidad más popular de Zotero es la generación automática de referencias bibliográficas. En el panel de la derecha encontrarás estos campos, que te servirán para que se generen automáticamente bibliografías en los diversos estilos de citación.

Por ejemplo, al exportar esta citación en formato APA, el resultado será el siguiente:

> Crews, K. (2008). Estudio sobre las limitaciones y excepciones al derecho de autor en beneficio de bibliotecas y archivos. Recuperado el 27 de enero de 2016, de World Intellectual Property Organization WIPO website: http://www.wipo.int/meetings/es/doc_details.jsp?doc_id=109192
>
> Cita realizada con el estilo APA

Esta misma referencia bibliográfica, exportada en formato Chicago quedará así:

> Crews, Kenneth. 2008. «Estudio sobre las limitaciones y excepciones al derecho de autor en beneficio de bibliotecas y archivos». World Intellectual Property Organization WIPO. 2008. http://www.wipo.int/meetings/es/doc_details.jsp?doc_id=109192



Panel derecho de Zotero, en la pestaña de Información

> Cita realizada con el estilo Chicago

Y en el caso del estilo Vancouver se ve así:

> Crews K. Estudio sobre las limitaciones y excepciones al derecho de autor en beneficio de bibliotecas y archivos [Internet]. World Intellectual Property Organization WIPO. 2008 [citado el 27 de enero de 2016]. Disponible en: http://www.wipo.int/meetings/es/doc_details.jsp?doc_id=109192
>
> Cita realizada con el estilo Vancouver

Para generar estos diferentes estilos, que a veces incluyen el nombre y apellido completo del autor, a veces sólo su apellido y la inicial de su nombre, el año de publicación a veces no aparece o a veces debe aparecer pero entre paréntesis, etc. sólo debí completar la mayor cantidad de campos posibles en Zotero y exportar esta referencia bibliográfica seleccionando el estilo de citación que necesito. Zotero automáticamente se ocupa de estos detalles.

Para nuestras infraestructuras de bolsillo usaremos la exportación en BibTex, lo que genera un archivo que podemos incluir mediante Pandoc a nuestro archivo en Markdown y desde ahí incluir las referencias automáticas en el formato al que queramos exportar nuestra publicación.

Crear o participar en grupos que comparten referencias bibliográficas

Algunas redes sociales están hechas para coleccionar amigos, otras para coleccionar seguidores. Otras se crearon para compartir fotos o para coleccionar contactos profesionales. Ya que hasta las mascotas tienen redes sociales no es de extrañar que exista una red social para compartir referencias bibliográficas y esa es Zotero.

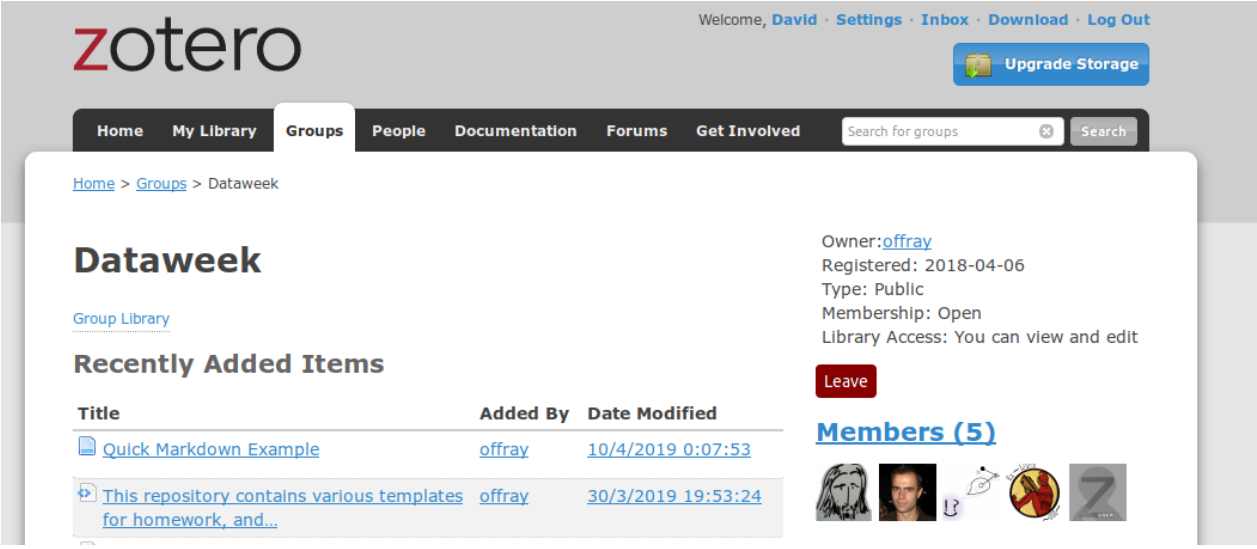
En el sitio web www.zotero.org podrás iniciar sesión o crear una cuenta. Por ejemplo en mi perfil de usuario de Zotero puedes ver *mis publicaciones*, es decir los documentos que he escrito, *mis contactos*, *mi biblioteca* donde se guardan las referencias que vimos en la sección anterior y *mis grupos*, donde puedes crear carpetas colaborativas donde mucha gente incluye referencias bibliográficas o gestionan para completar sus datos a muchas manos.

Los grupos

En los grupos tendras carpetas, pero que son gestionados por múltiples usuarios. Por ejemplo en la siguiente imagen, puede verse el grupo del Dataweek desde Zotero.

Fíjate que en la imagen en el panel izquierdo, en la parte superior aparece «Mi biblioteca», que es esa carpeta donde se guardan las referencias bibliográficas personales, que vimos en la sección pasada. Un poco más abajo encontrarás «Bibliotecas de grupo» que contiene los diferentes grupos a los que perteneces en Zotero.

Desde el sitio del grupo de Dataweek en Zotero puede verse la fecha de creación del grupo, las personas que lo conforman y desde allí puedes incluirte en el grupo. A inicios de 2019 cuando escribimos este libro el grupo cuenta con 5 integrantes, lo que quiere decir que cada una de estas 5 personas puede aportar referencias bibliográficas, crear o editar carpetas y subcarpetas o añadir notas, etiquetas e incluso archivos adjuntos a las referencias bibliográficas.



Crea tu grupo

Cuando hayas creado tu cuenta en Zotero puedes ir a la sección de Grupos donde te aparecerá la opción para buscar grupos o crear tu grupo. Si creas tu grupo deberás elegir su nombre y el tipo de acceso que tendrá:

- Público, con membresía abierta.
- Público, con aprobación para integrar el grupo o membresía cerrada
- Privado, sólo se puede participar con invitación.

Ahora ya sabes cómo crear un grupo, pero si estas leyendo este libro te invitamos a que hagas parte del Grupo del Dataweek en Zotero. donde varios de los integrantes de la Comunidad de Grafoscopio participan.

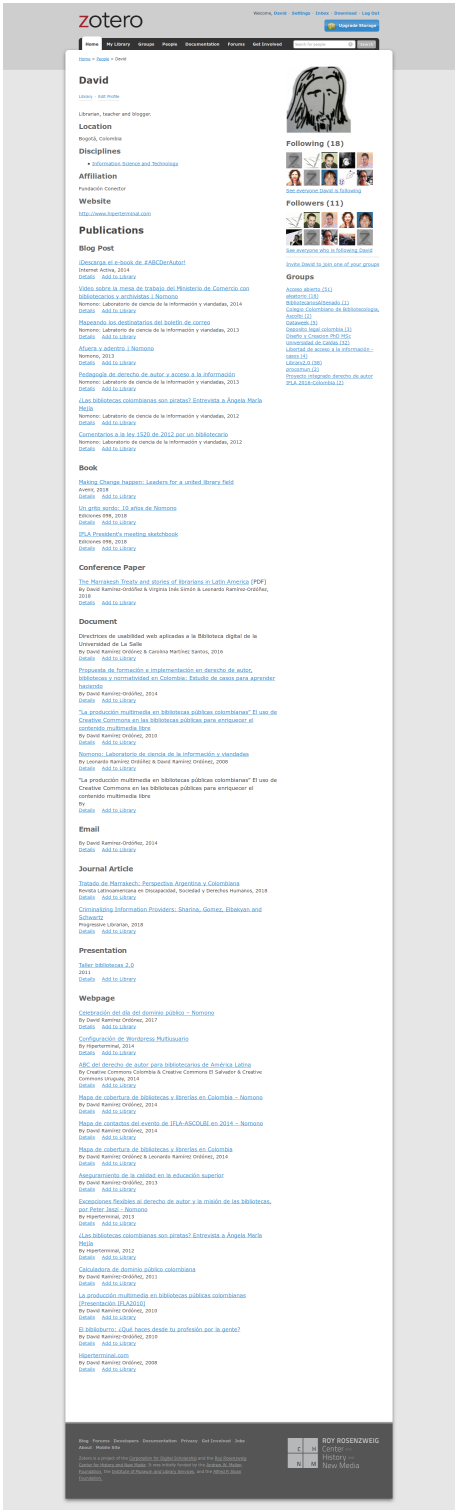
Compartiendo referencias bibliográficas: optimizando tiempo y esfuerzos

Como lo comenté en la sección **Gestionar sus referencias bibliográficas personales**, una de las propiedades de esta herramienta es que sólo necesitas escribir una referencia bibliográfica una vez e incluirla en cuantas carpetas consideres que pueden aportar para crear una bibliografía.

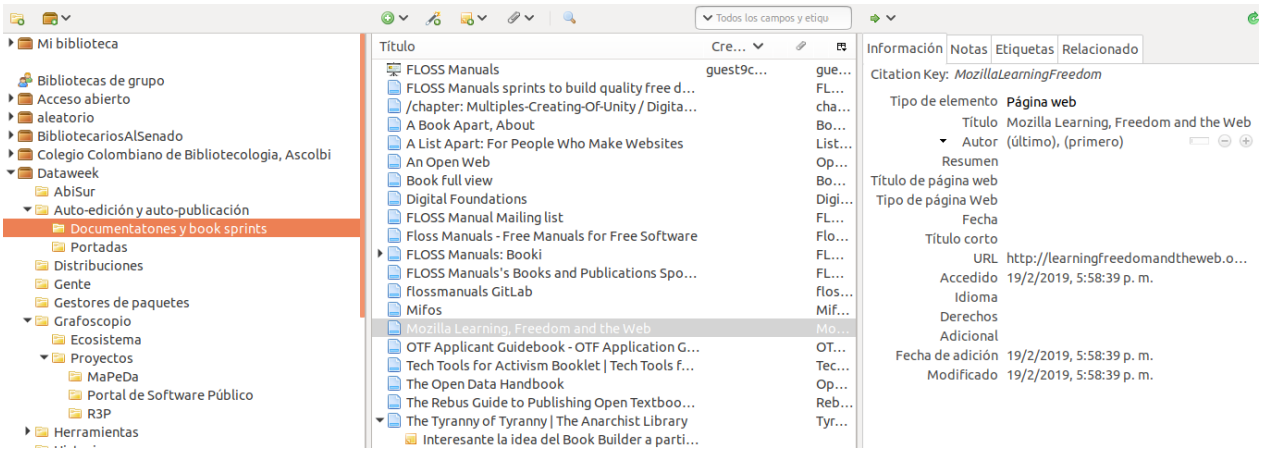
Debes ser cuidadoso porque las referencias guardadas en los grupos no se sincronizan con las referencias guardadas en el espacio personal. Esto quiere decir que si creasta una referencia en tu espacio personal y

la compartiste en un grupo de Zotero, las dos referencias, a pesar de ser del mismo documento, pueden modificarse y *desarrollarse* de forma diferente. Por ejemplo alguien pudo haber añadido más datos a la referencia grupal y por tu parte incluiste notas, etiquetas y archivos adjuntos a la referencia en tu espacio personal. Al final podrías tener grandes diferencias entre una y otra.

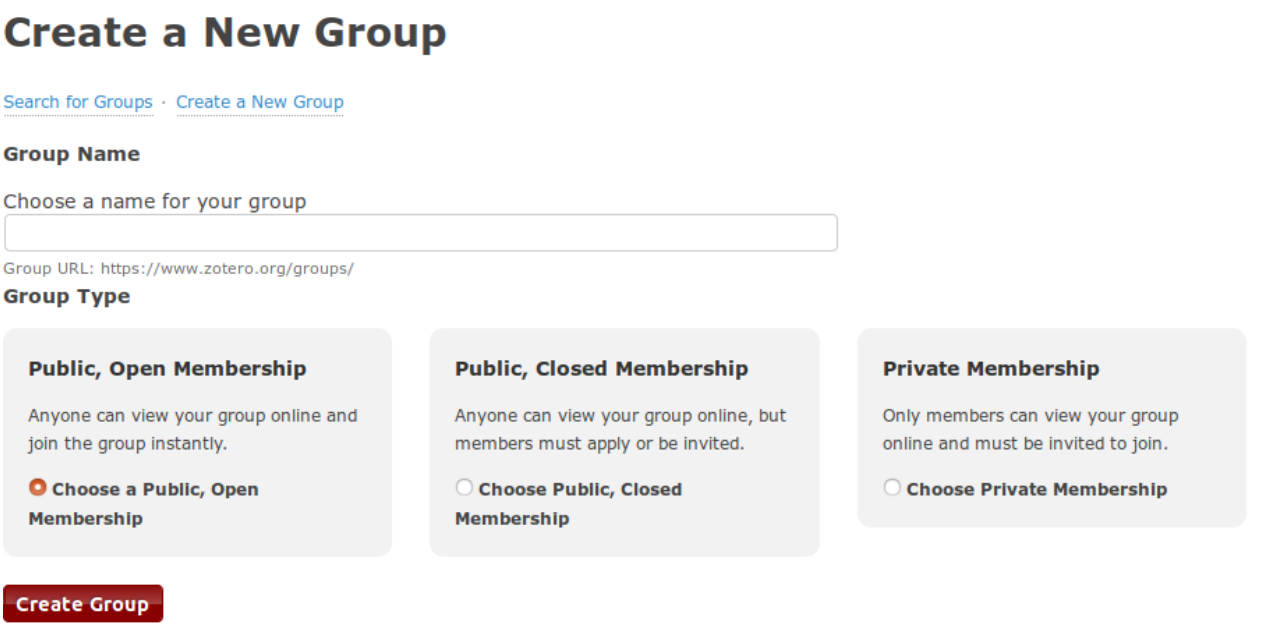
Como ves Zotero es una infraestructura de bolsillo que hace parte de Grafoscopio, que nos permite compartir referencias a libros, sitios web, videos y demás para leer conjuntamente. En la biblioteca del Grupo de Dataweek ya tenemos más de 270 documentos relacionados con la visualización de datos, la edición digital y la escritura colaborativa.



Captura de pantalla de un usuario de Zotero



Captura de pantalla de Zotero, mostrando el grupo de Dataweek



Creación de grupos en Zotero

Licenciamiento

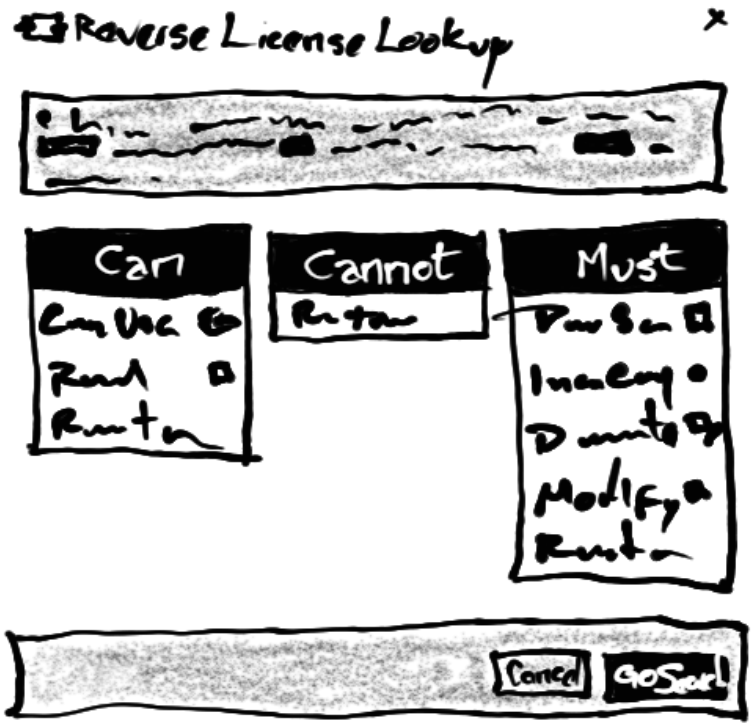


Imagen alusiva a la selección de licencias

La licencia es la explicitación de los acuerdos que nos importan (o no). Desafortunadamente, si no elegimos una licencia explícitamente, el sistema de derecho de autor lo hará por nosotros y usualmente elegirá algo extremadamente restrictivo, que puede ser muy útil para los monopolios de distribución de contenidos,

pero poco para autores, creadores, lectores y la cultura en general. Por eso es conveniente elegir y establecer la licencia por omisión para nuestras obras.

Hay varios sitios para escoger licencias, pero TL;DR Legal (Too Long; Didn't Read Legal) da una visión panorámica y detallada a la vez, que los promotores de licencias particulares (ejp Creative Commons) no ofrecen.

Un espectro de derechos

Acá va una explicación del copyright, el copyjust right, el copyleft, el copy far left y el copy south.

Como elegir una licencia

Creamos un pad y unas convenciones para evaluar los atributos que queríamos para nuestra obra, de modo que en base a ellos escogiéramos una licencia.

Estas fueron las convenciones:

- **MI**: Muy Importante
- **I**: Importante
- **N**: Neutral
- **PI**: Poco Importante
- **NI**: Nada Importante
- **C**: Condicionado

E hicimos una tabla con una escala de Licklider modificada, donde colocamos nuestros votos y argumentamos públicamente en favor de ellos. Este fue el resultado:

Libertad	MI	I	N	PI	NI	C
Relicenciamiento	+					++
Comercializar	+					++
Acceso al código Fuente	+++					
Compartir Igual	+		+			+

Libertad	MI	I	N	PI	NI	C
Copia	+++					
Atribución	+++					

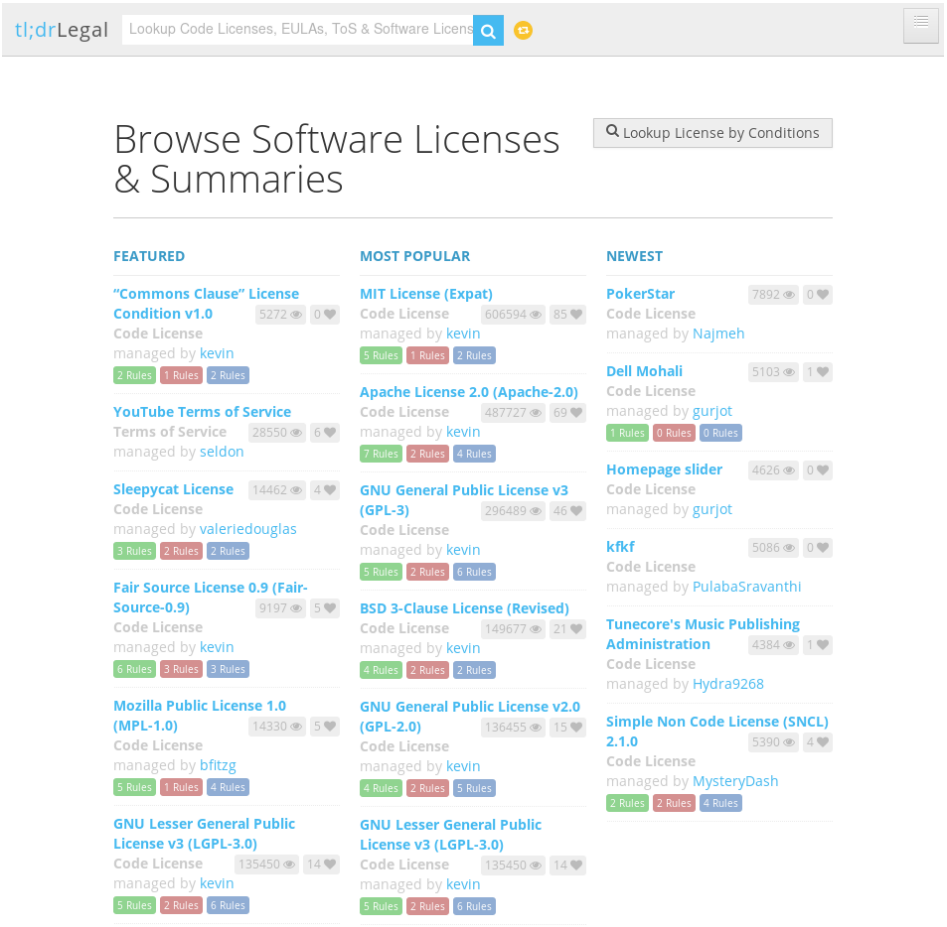
Estas son las licencias candidatas:

- ☐ Dominio Público.
- ☐ Gnu Free Documentación.
- ☐ Creative Commons Atribución Compartir Igual.
- ☐ P2P License.

  - Michael Bauwens: P2P Foundation.
  - Dmitri Kleiner: Manifesto Telecomunista.

- ☐ Academic Free License

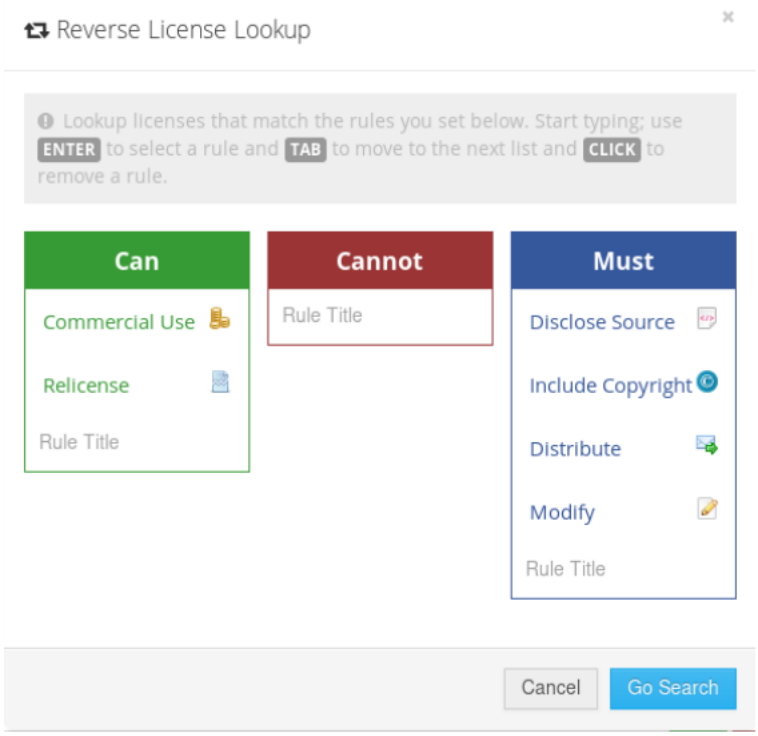
Para licenciar entramos a TL;DR Legal Browse Licenses



Y elegimos el botón en la esquina superior derecha el botón que dice «Lookup License by Conditions» y expresamos los requerimiento de nuestra tabla en el buscador reverso de licencias, lo que se vería de esta forma:

No encontramos nada que contenga esas libertadas, pero encontramos algo con libertades similares:

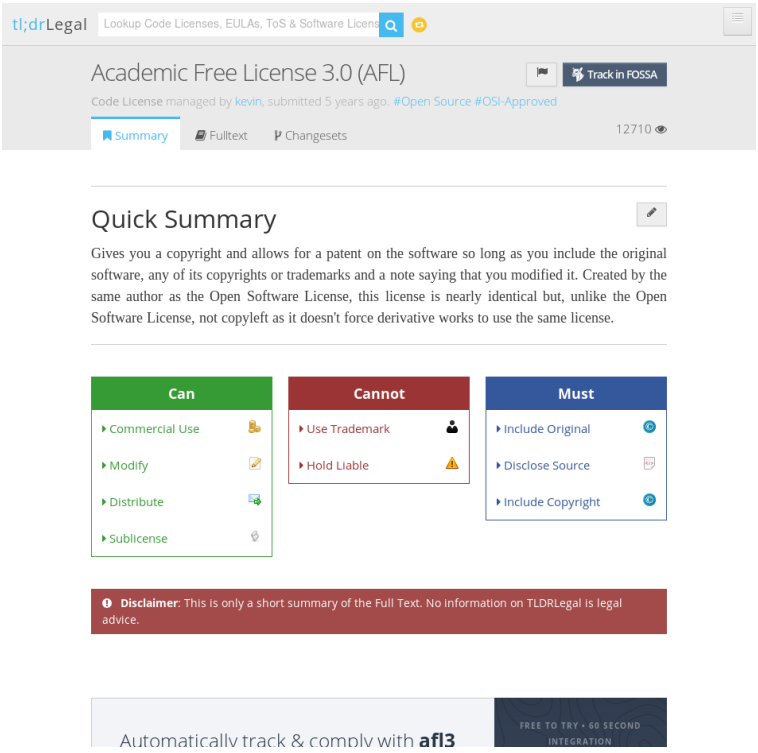
Podríamos desarrollar nuestra propia licencia, porque nos interesa mucho poder acceder al código fuente, pero no hay muchas licencias que permitn esas cosas. Por el momento no vamos a redactar nuestra propia licencia, por lo que usaremos una Academic Free License y condiciones al repositorio para que se favorezca el relicenciamiento. Así, quienes contribuyan al repositorio podrían relicenciar y luego colocaremos términos y condiciones de uso al repositorio para esto.



Selector reverso de licencias

Cómo leer una licencia

Acá viene una explicación del espectro de licenciamiento, desde el copyright hasta el copyfar left y las libertades que configuran dicho espectro.



La AFL (Academic Free License) versión 3 cumple con los requerimientos que queremos para nuestros archivos.

Escribir y publicar juntos

Instaladores y gestores de Software

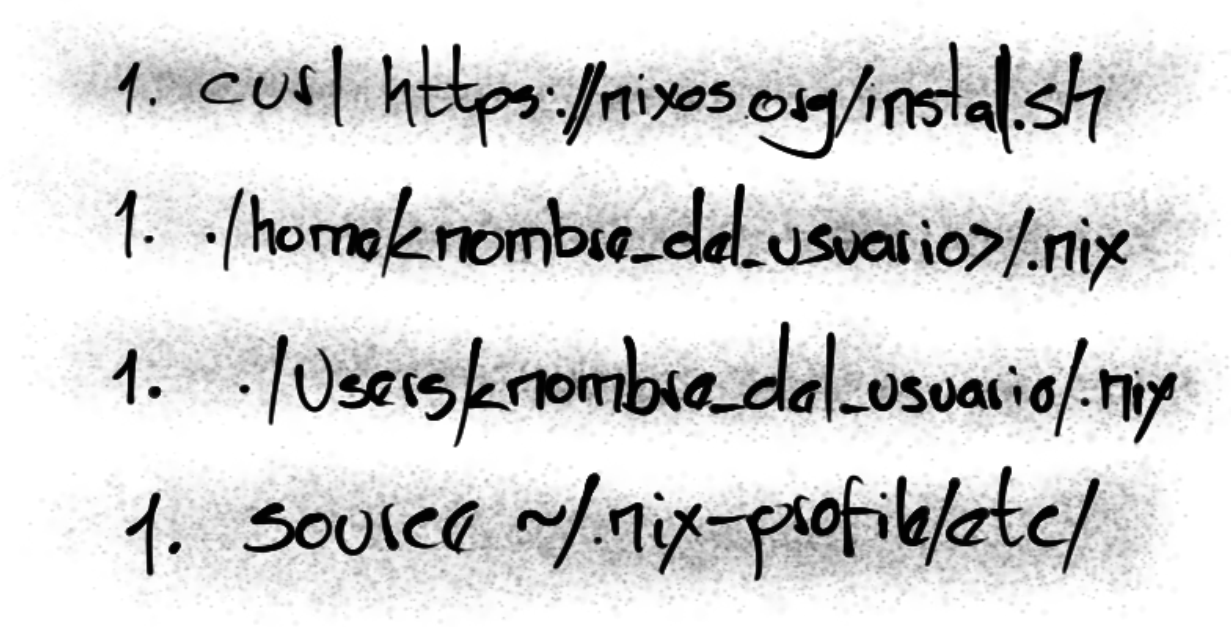


Imagen alusiva a instalación de software desde consola

Los gestores e instaladores de software, a pesar de no ser igual, se aplican al problema de adquirir software nuevo de maneras consistentes y replicables y también de desistalarlos.

Estos sistemas permiten realizar lo que se conoce como instalaciones desatendidas, es decir instalaciones

automatizadas de software en las que la persona sólo menciona el paquete que quiere (ejp: flash) y la operación que va a realizar (instalar, actualizar, desinstalar) y el gestor de paquetes se encarga de realizar esta tarea sin mayores intervenciones

Es buena idea antes de instalar o configurar los instaladores/gestores que se verifiquen si ya se encuentran instalados. Una forma de verificar es mirar si por ejemplo los comandos `nix` o `brew` ya existen y a su vez verificar que programas específicos como `fossil` o `pandoc` ya estén instalados antes de intentar instalarlos por medios alternativos.

Windows

En esta sección vamos a considerar dos sistemas para Windows.

Scoop

Sitio web de Scoop

Para instalar Scoop son tres sencillos pasos:

- Abrir Power Shell:
- (Opcioal): Habilitar la política de ejecución para la instalación:

```
Set-ExecutionPolicy RemoteSigned -scope CurrentUser
```

- Instalar Scoop:

```
iex (new-object net.webclient).downloadstring('https://get.scoop.sh')
```

Chocolatey

Sitio web de Chocolatey

Mac

Homebrew/brew

https://brew.sh/

Instalar

Para instalar hay que ejecutar el comando

```
1 /usr/bin/ruby -e "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/
    install/master/install)"
```

Buscar un paquete

Buscar el paquet fossil

```
1 brew search fossil
```

y para instalar el paquete

```
1 brew install fossil
```

Y hacer lo mismo con pandoc en lugar de fossil.

Gnu/Linux (y Mac)

Las distribuciones de Linux cuentan con sus propios gestores de paquetes y es allí donde la idea más maduró y pasó a otros sistemas operativos. Sin embargo, en ocasiones el gestor nativo no siempre trae los paquetes que necesitamos, o las versiones no son las requeridas y en ese sentido es conveniente contar con otros gestores.

Nix

Nix es un gestor de paquetes alternativo para sistemas tipo Unix, lo cual incluye distintas variantes de Gnu/Linux y Mac.

GuixSD: Es una alternativa que están inspiradas en Nix, pero usan un único lenguaje de programación (Guile), en lugar de varios, como Nix.

Instalación

Abrimos la terminal o consola (se llama diferente dependiendo del sistema operativo, pero la funcionalidad es la misma), que nos permite ejecutar comandos en dichos sistemas y escribimos el siguiente comando de instalación:

```
curl https://nixos.org/nix/install | sh
```

Para habilitarlo, desde la consola de comandos teclamos:

```
. /home/<nombre_del_usuario>/.nix-profile/etc/profile.d/nix.sh
```

Para MacOS

```
    . /Users/<nombre_del_usuario>/.nix-profile/etc/profile.d/nix.sh
```

Forma alternativa para habilitarlo (si la anterior falla):

```
source ~/.nix-profile/etc/profile.d/nix.sh
```

Una vez instalado, podemos consultar la base de datos de todos los paquetes (tarda un poco la primera vez)

```
nix-env -qa
```

Buscar un paquete en particular:

```
nix-env -qa | grep fossil
```

Instalar un paquete específico:

```
    nix-env -i fossil
```

Varios

- Opcional: consola de Ubuntu (bash/apt) en Windows (195 MB) https://www.microsoft.com/es-es/store/p/ubuntu/9nblggh4msv6#system-requirements
- Instalar Acrobat Reader para Windows vía Wine.

Markdown: Documentación ágil y estructurada

Imagen alusiva al logo de Markdown

Introducción

Las pantallas se están volviendo más ubicuas y diversas. Tabletas, teléfonos celulares, computadores portátiles y de escritorio y los libros son aparatos donde expresamos y nos aproximamos a ideas, causas e historias. Esto configura un paisaje de medios donde estamos leyendo y escribiendo desde y para distintos aparatos y factores de forma. En tal panorama diverso, software popular, como las *suites* de oficina, pueden ser consumir demasiados recursos y ser extra complicadas, haciéndose inadecuadas para escritura colaborativa y/o multi-dispositivo, y esto crea una creciente tendencia a explorar otras maneras de escribir, que recobran el foco en el núcleo de la experiencia de escritura: poner palabras juntas, mientras que proveen un formato sencillo y una experiencia de usuario minimalista que aún refleja la estructura del documento y su presentación. Es por eso que hablamos acá de **documentación ágil y estructurada**: estas técnicas no son voraces en recursos (cognitivos o tecnológicos), pero proveen los elementos necesarios para escribir documentos estructurados jerárquicos (si es necesario), con secciones, subsecciones, capítulos, subcapítulos etc y también soportan otras formas de estructuras, como notaciones musicales o matemáticas, entre otras. Acá exploraremos uno de esas maneras ligeras de escritura, así como las herramientas para ella.

:point_right: **Usa Markdown si**:

- Quieres escribir colectivamente mientras mantienes la colaboración y la escritura simple:
    - en un encuentro tipo maratón o carrera de escritura.
    - en un equipo distribuido de colaboradores distribuidos en varias zonas horarias y ubicaciones.
- Si quieres escribir desde/hacia varias pantallas y factores de forma, mientras quieres mantener un único archivo fuente para producir múltiples salidas: impresa (PDF), web (HTML) y móvil (EPUB).

El lenguaje de etiquetado Markdown es una herramienta que permite a los autores escribir desde diversos dispositivos, produciendo, de manera sencilla un documento que puede ser exportado a diversos formatos, dispositivos de medios y pantallas: impresa (PDF), móvil (EPUB) y web (HTML). Y, a pesar de la facilidad de Markdown, esto no es un impedimiento para crear complejos documentos, como libros completos. Pero puedes aprenderlo de una manera orgánica, yendo de las bases a más complejas configuraciones y casos de uso. Este conjunto de características hacen también de Markdown ideal para ejercicios y contextos de documentación orgánica y colaborativa, como talleres, maratones de escritura o el trabajo con colaboradores repartidos a través de distintas zonas horarias y lugares, porque trazar los cambios y hacer comentarios puede convertirse en algo realmente simple, cuando un formato simples es la base para la escritura (esta es la clave de inmensas obras colaborativas, como la Wikipedia, por ejemplo, pero también puede ser usado en manuales, libros, entre otros, como acá veremos).

Por ejemplo, para crear una memoria colaborativa de los talleres y hackatones en nuestro hackerspace local, usualmente creamos un texto colaborativo en un editor llamado CodiMD y empezamos a escribir justo ahí, lo cual facilita hacer ediciones y mejoras a dicha memoria, sin perder la agilidad en el proceso. De hecho, mucha de la documentación que estás viendo en esta obra, fue bosquejada y escrita usando un formato y herramientas tan sencillas (en comparación con sus contrapartes de la ofimática clásica).

Markdown es quizás el lenguaje de etiquetado más popular para documentación usado por millones de personas en diferentes comunidades, como aquellas que se congregan en StackOverflow, GitHub y GitLab.

:heavy_exclamation_mark: Existen numerosas variantes de Markdown que comparten lo básico, pero son extendidas para las necesidades de comunidades particulares. Así, cuando estás aprendiendo Markdown, una vez hayas cubierto lo esencial, aprenderás alguna variante particular (o muchas).

Esta parte de la obra te proveerá con los enlaces sobre cómo usar dicha Markdown y los recuros que puedes encontrar para profundizar en este camino. Debido a que hay muchos recursos en línea sobre éste, más que repetirlos acá, lo que haremos será proveerte con un tour guiado sobre cómo estos recursos están organizados para ir de lo simple a lo complejo.

Herramientas y ejemplos relacionados

Esta sección muestra algunas extensiones y variantes de Markdown, ejemplos de obras particulares y también herramientas que pueden ser usadas para escribir en él.

- (10:44) Mira Academic Writing in Markdown para obtener una panorámica de las razones y procesos detrás de usar este lenguaje de etiquetado (la mayoría de ellos aún aplica fuera del contexto académico).
- Critic Markdown: Una variante de Markdown para trabajo colaborativo entre autores y editores, que soporta resaltado de insersiones, borrados, substituciones y comentarios.
- Pandoc: Una herramienta que permite y importar y exportar desde/hacia varios lenguajes de etiquetado, incluyendo diveras variantes de Markdown. Tiene su propia sintaxis extendida de Markdown para lidiar con varios requerimientos de la publicación de libros y puede combinar lenguajes de etiquetado más complejos (como LaTeX) para crear sofisticadas publicaciones, como libros, tesis y revistas, entre otras.
- Libro de periodismo de datos, la versión de código abierto: Muestra cómo el Markdown de Pandoc puede ser usado para recrear un libro completo en formato PDF, incluyendo su diagramación, gráficas y tipografías.

- Zettler: Un editor de Markdown que soporta previsualización en caliente.
- CodiMD: Un editor colaborativo para Markdown con resaltado sintáctico, previsulización en vivo permisos granulares sobre los documentos y soporte para varias extensiones de Markdown como bloques de metadatos en Yaml, *smileys*, bloques de alerta y trozos de código, entre otros.
- Markor: Un editor de texto para Android basado en Markdown para notas, agenda y enlaces.

Aprendiendo los fundamentos

Estos pasos te permitirán aprender las bases de Markdown de una manera progresiva. Hay muchos recursos para aprenderlo en línea, así que no vamos a repetirlos acá. Sólo vamos a introducir algunas modificaciones mínimas para mejorar la experiencia de aprendizaje. Algunas de las experiencias estarán relacionadas con leer, mientras otras requerirán hacer. Recuerda tener en cuenta las convenciones.

- Familiarizate con la sintaxis, empezando con el Tutorial de Markdown.
- Lee del *Historiador que Programa* Introducción a Markdown para una introducción más comprenhensiva.
- Prepara el software para trabajar con Markdown:
    - Para su edición:
        - Crea una cuenta de CodiMD para trabajar desde tu navegador web en el servidor de demo, o, mejor aún créala en nuestro propio servidor comunitario.
        - Instala Zettlr, para trabajar fuera de línea en tu propio computador portátil o de escritorio.
    - Para conversión de formatos instala Pandoc.
- Sigue la Escritura sostenible en texto plano usando Pandoc y Markdown de el Historiador que Programa, para crear tu primer documento, pero usa CodiMD or Zettlr en lugar del editor de texto sugerido allí.

Metadatos

Lo metadatos son datos sobre los datos. Varios sistemas de escritura basados en Markdown usan un lenguaje de metadatos ligero, llamado YAML, que sirve para controlar tanto las opciones del documento general, como las de cada uno de los subdocumentos que lo componen. Para el primer caso, se usa un archivo titulado `metadata.yaml`, que veremos más adelante y en el segundo, se emplea algo llamado el

bloque de metadatos yaml, que se puede colocar al comienzo de cada documento en Markdown entre tres guiones sencillos (`---`).

YAML define los documentos de manera arbórea, es decir, como una jerarquía de datos que puede contener a otros datos. Cada dato a su vez es un par `atributo: valor`, donde lo que está antes de los dos puntos, es el nombre del atributo y lo que está después es el valor del mismo. Así `titulo: "Documentatón"`, nos indica que en nuestros metadatos hay algo llamado `titulo` y que el valor del mismo es `"Documentatón"`.

Para una introducción más detallada sobre YAML se recomienda ver:

- Página de YAML en la Wikipedia en español.
- Página de YAML en la Wikipedia en inglés.
- https://yaml.org/: Especificación técnica del estandar YAML e implementaciones que lo soportan en varios lenguajes de programación.

Una ventaja de YAML es que permite definir los metadatos de manera emergente, en la medida en que los vamos necesitando, pues basta con definir un par `atributo: valor`en la cabecera de un documento, para poder hacerle consultas u operaciones a ese documento de acuerdo a los atributos que definimos. Por ejemplo, encontrar todos los documentos cuya nivel (`level:`) sea principiante (`beginner`) o buscar todos los atributos en los metadados de un documento particular. O usar los metadatos para definir el tamaño de fuente o la plantilla con la cual exportaremos el documento a distintos formatos como PDF o EPUB. Otra de las ventajas de los metadatados, es que si los usamos consistentemente, podremos recuparar una obra completa a partir de uno de sus fragmentos, pues dicho trozo puede contener un metadato que permita encontrar la obra completa.

Es importante, sin embargo, que una vez hayamos definido los metadatos de un documento, los usemos de manera consistente en otros documentos que hacen parte de la misma colección. En el caso de los documentos para esta obra hemos definido los siguientes metadatos. Cada uno va antecedido con un comentario en YAML que explica brevemente su propósito y si hay varias opciones exclusivas (para seleccionar una) estas van colocadas entre paréntesis y separadas por una barra vertical (`|`).

```
1 # Controla los saltos de línea (true | false)
2 breaks: false
3 # Define el repositorio de código al que pertenece el documento.
4 repo: enlace
5 # Define si un documento es una subsección de otro documento.
6 parent: archivo.md
7 # Define el listado de enlaces donde este documento se replica.
```

```
sync:
  - enlace1
  - enlace2
# Define traducciones de un mismo documento. Los idiomas deben
# definirse usando el código de dos letras ISO 639-1
translations:
  - idioma1: enlace1
  - idioma2: enlace2
# Define el nivel de aprendizaje para el que está pensando el documento:
level: (beginner|intermediate|expert)
```

CodiMD: Escritura síncrona colectiva

En ocasiones queremos escribir a muchas manos. Puede ser que estemos en un taller y queramos tomar notas de manera colectiva. O bien estamos trabajando en un documento y quisiéramos reunirnos con coautores para arrojar ideas, escribir subsecciones o realizar correcciones. Para ello se han creado varias plataformas que con sólo un enlace web permiten iniciar la colaboración con todos los que tengan dicho enlace (suponiendo que tienen los permisos para ello). Debido a que cuando los coautores se encuentran a través de la web, pueden escribir al mismo tiempo y ver los cambios de otros, se dice que dichas plataformas trabajan en «tiempo real».

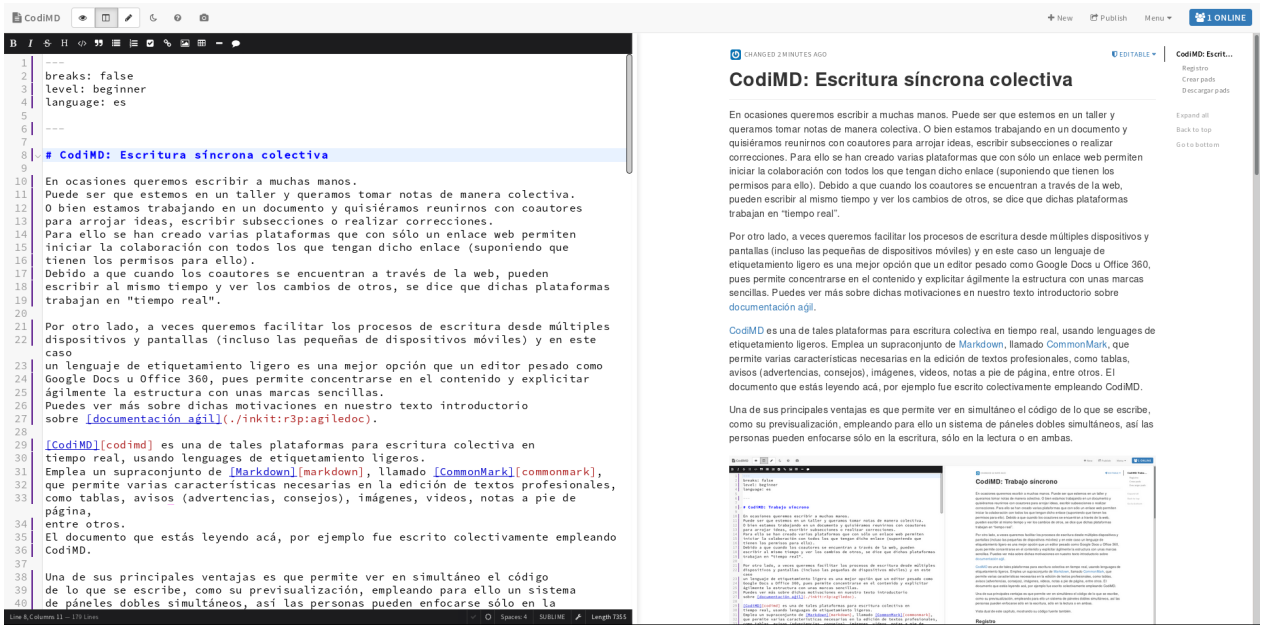
Por otro lado, a veces queremos facilitar los procesos de escritura desde múltiples dispositivos y pantallas (incluso las pequeñas de dispositivos móviles) y en este caso un lenguaje de etiquetamiento ligero es una mejor opción que un editor pesado como Google Docs u Office 360, pues permite concentrarse en el contenido y explicitar ágilmente la estructura con unas marcas sencillas. Puedes ver más sobre dichas motivaciones en nuestro texto introductorio sobre documentación ágil.

CodiMD es una de tales plataformas para escritura colectiva en tiempo real, usando lenguages de etiquetamiento ligeros. Emplea un supraconjunto de Markdown, llamado CommonMark, que permite varias características necesarias en la edición de textos profesionales, como tablas, avisos (advertencias, consejos), imágenes, videos, notas a pie de página, entre otros. El documento que estás leyendo acá, por ejemplo fue escrito colectivamente empleando CodiMD.

Una de sus principales ventajas es que permite ver en simultáneo el código de lo que se escribe, como su previsualización, empleando para ello un sistema de páneles dobles simultáneos, así las personas pueden enfocarse sólo en la escritura, sólo en la lectura o en ambas.

Registro

El registro se puede hacer de dos formas, usando herramientas externas como GitHub, Google o Twitter o a través del registro por correo electrónico.



Vista dual de este capítulo en CodiMD, mostrando su código fuente también.

Revisar definición de las capturas

Luego de elegir una de estas opciones y validar el registro, es posible editar los documentos.

A diferencia de los Pads que se han venido utizando en los Data Rodas anteriores, en donde los documentos quedan públicos y abiertos a la edición de cualquier persona, en esta oportunidad se ponen en debate la limitacion en la edición de contenidos por terceras personas agenas al proyecto, con el fin de garantizar la privacidad y seguridad de la documentación producida.

Regístrate en un proveedor de CodiMD como Docutopia o Heroku.

Crear pads

Escencialmente hay dos formas:

- Desde la portada de la instancia de CodiMD.

  A través del botón azul en la esquina superior derecha de la pantalla, como se vé a continuación

- A través de un enlace al nuevo documento.

  Durante el proceso de documentación colaborativa podemos incluir enlaces en los documetos de manera fluida, para los efectos de esta comunidad se van a relacionar contenidos siguiedo la estructura de enlaces de Markdown, con la siguiente convención:

  - se incluirá entre corchetes cuadrados el texto que va a contener el enlace, por ejemplo `[texto]`.
  - Luego se incluirá el complemento de la URL del recurso, determinando la ruta de orignen del nuevo documento entre paréntesis «`()`».
  - El contenido entre paréntesis incluirá la ruta a partir del «./» para referirse al servidor que estamos usando, seguido de dos puntos (`:`) para diferenciar las «carpetas» o secciones del tema relacionadas con el contenido.
  - Finalmente, el resultado será algo como texto. Para referenciarlo al ejercicio de esta sesión sería algo como

    ```
    1 [texto](./tema:subtema:pad)
    ```

    En donde el texto del enlace «`texto`» refiere al lugar usando en el servidor de Docutopia relacionado con los contenidos de la tema y el subtema.

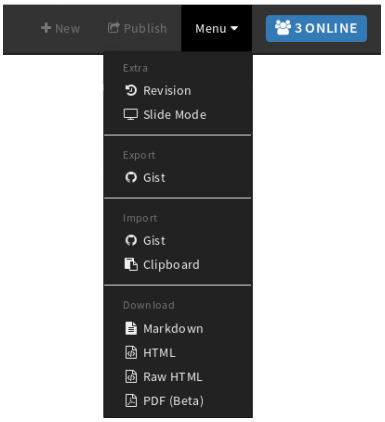
Crea un pad en el proveedor en el que te registraste en la actividad previa.

Descargar pads

Una vez hemos creado nuestros documentos en línea, puede ser necesario descargarlos para tener una copia de respaldo de los mismos o trabajarlos fuera de línea. Acá veremos tres métodos para ello.

Desde el navegador

A través de la vista de escritura, podemos acceder al menú desde el botón en el margen superior derecho, donde se muestran las diferentes opciones de descarga del documento: Markdown, HTML, HTML Raw y PDF.



Menú de CodiMD, con las opciones de descarga

Descarga un pad desde el navegador.

Con wget

Suponemos que para esta parte tienes instalada la terminal o consola, o línea de comandos, que sabes cómo abrirla y los elementos básicos de escribir y ejecutar comandos en ella. También que tienes instalados los programas que se usan habitualmente en ella, como `wget`.

Usaremos los términos terminal, consola o línea de comandos de manera intercambiable.

Otra de las maneras de descargar un documento es emplear la terminal. La ventaja de este procedimiento es que podemos agilizar muchas descargas con sólo repetir un comando de una línea, lo cual abre a su vez la posibilidad de automatizarlo.

Para descargar este documento, abre la terminal y escribe:

```
1   wget -O codimd.md https://docutopia.tupale.co/inkit:r3p:codimd/download
```

Con esto hemos creado un archivo llamado `codi.md`, en el disco duro, con los contenidos guardados en la dirección del pad cuyo enlace es `inkit:r3p:codimd` y está hospedado dentro del servidor `https://docutopia.tupale.co`. Si queremos descargar otros pads con otros enlaces, creados en otro servidor, o guardarlos con otros nombres de archivo, deberemos cambiar el comando previo respectivamente.

Descarga un pad utilizando `wget`.

Con Pharo

Pharo es la plataforma sobre la cual está hecho Grafoscopio, que a su vez sirve para integrar y automatizar mucho del trabajo sobre investigación y publicación reproducible y remezclable, así que veremos cómo emplearla en las actividades de este texto de maneras progresivamente más elaboradas.

Pistas y referencias extra:

- Elegant code.
- CodiMD CLI

Zettlr: Un editor de Markdown fuera de línea

Markdown no sólo sirve en editores conectados a Internet, como CodiMD donde no sólo escribes en Markdown, sino que lo haces a múltiples manos colaborando en tiempo real. Se puede usar un bloc de notas para escribir markdown en tu computador personal o teléfono móvil, pero aquí revisaremos Zettlr, un bloc de notas con súper poderes que puedes instalar en tu computador y tener una experiencia de escritura mejorada.

En este capítulo veremos:

- Una breve comparativa de algunos editores de markdown.
- Un manual de uso de Zettlr.

Editores de markdown

Antes de llegar a Zettlr probé estos editores:

- Atom
- Geany
- Ghostwriter
- Sublime Text

Si bien pueden haber muchos más editores de Markdown, probé estos y aquí una breve reseña de lo que me gustó y lo que no.

Atom

Atom es un editor de texto increíble. Tiene muchísimas funcionalidades, pero tal vez por eso mismo es bastante pesado. Uso un computador que se parece más a una máquina de escribir porque la uso sobre todo para labores que tienen que ver con procesar texto como escribir documentos, leer y enviar correos,

publicar en mi blog, pero no hago cosas como edición de video que demandan muchos recursos del equipo. Atom se demoraba mucho en cargar y por eso lo descarté.

Sin embargo tiene funciones muy interesantes como Teletype, que permite conectar este procesador de texto con un repositorio en línea, combinando la edición síncrona con la asíncrona en una sola herramienta. Otro paquete que me gustó de Atom fue Zotero-picker, que te permite incluir referencias bibliográficas en Zotero a los textos que escribes en markdown.

Atom permite previsualizar los textos escritos en markdown en *html*, lo que te puede ir dando una guía de cómo va quedando lo que escribes.

Geany

Geany es un editor de texto bastante liviano, pero sentía que no estaba hecho para markdown. En él suelo editar *html* y otros archivos de código. Podría funcionar bien para markdown, pero seguí explorando otras opciones.

Ghostwriter

Ghostwriter es uno de los que más me ha gustado porque está hecho para escribir en markdown, donde a medida que vas usando las diferentes características de markdown, como títulos, citas, negrillas, etc. el texto se va ajustando para ofrecer una previsualización de cómo va quedando el texto.

Tiene otras funcionalidades como la escritura sin distracción, que hace que lo que esribes ocupe toda la pantalla y no tengas otros botones, menús o cosas que te distraigan. En estos tiempos de la economía de la atención, me pareció que escribir en Ghostwriter es una experiencia bastante limpia.

También cuenta, al igual que Atom, con una previsualización en *html*, así ves cómo queda lo que escribes. Tiene una funcionalidad que a pesar de ser sencilla, me pareció muy interesante y es el *modo Hemingway*. Cuando activas este modo, no puedes devolverte o borrar lo que escribes. El software te obliga a que escribas de un solo tirón todas las ideas que tengas en la cabeza para luego desactivarlo y si volver a corregir. Un *detalle de fina coquetería* como se diría en Bogotá.

De esta herramienta me gusta que es bastante liviana y automáticamente crea un backup de tu texto. Junto a tu archivo en markdown (es decir el nombre-archivo.md) crea un archivo.md.backup.

Sublime Text

Sublime Text es una herramienta muy buena para editar datos. Recuerdo mucho la experiencia de uso cuando tenía que buscar, reemplazar o editar texto ¡Es muy bueno! Sin embargo la cosa es que no es software libre, por lo que mucho de su encanto se pierde al no permitirte las libertades básicas de uso, copia, modificación y distribución.

Zettlr

Lo que me gustó de Zettlr es que es liviano y está hecho para markdown como Ghostwriter, tiene múltiples funcionalidades como Atom y no tiene la licencia de Sublime Text, ya que es software libre, usando una licencia GNU/GPL. Tiene unos conceptos o metáforas poco intuitivas, pero una vez que captas de qué va la idea del desarrollador, escribir fluye muy bien. De hecho mi proceso de escritura para este libro es redactar los textos en Zettlr, copiar lo que aquí redacto a CodiMD y hacer *commits* al repositorio de Fossil para que la *Comunidad de Grafoscopio y Dataweek* con quienes escribimos este texto acceda a los archivos fuente en sus últimas versiones.

Pero vamos desde el inicio: la instalación.

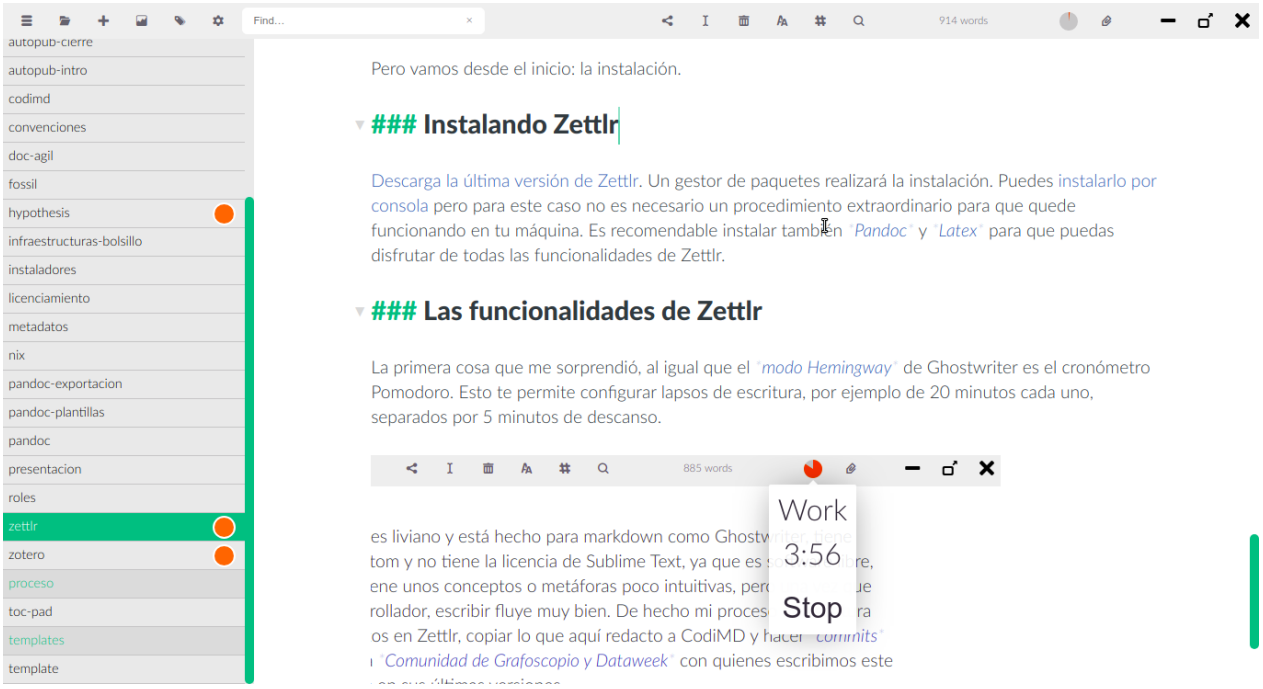
Instalando Zettlr

Descarga la última versión de Zettlr. Un gestor de paquetes realizará la instalación. Puedes instalarlo por consola pero para este caso no es necesario un procedimiento extraordinario para que quede funcionando en tu máquina. Es recomendable instalar también *Pandoc* y *Latex* para que puedas disfrutar de todas las funcionalidades de Zettlr.

Las funcionalidades de Zettlr

La interfaz

La interfaz de Zettlr es bastante limpia y al estar hecha para markdown, tiene una previsualización que hace que puedas previsualizar cómo quedará tu documento con el formato como títulos, itálicas, imágenes, enlaces y demás.



Captura de pantalla de la interfaz gráfica de Zettlr

Zettlr usa la metáfora de *directorios* y de *proyectos* para los archivos que aparecen en la columna izquierda. En esta herramienta se crean directorios en donde se almacenan archivos y si quieres abrir un archivo, primero deberás haber configurado el directorio en Zettlr para que lo reconozca. Eso es lo que no parece muy intuitivo, pero de resto todo es bastante minimalista.

Dentro de los directorios hay archivos y Zettlr te mostrará únicamente los archivos de markdown, es decir los .md, de un directorio específico. Así tendras acceso muy fácilmente a los diversos archivos que estés redactando. Por ejemplo en la figura anterior, configuré un directorio de la Documentatón, de tal forma que me muestre todos los archivos que se encuentran en mi computador y son copia del repositorio de Fossil de la Documentatón.

Cuando usas Zettlr, puedes usar el modo de escritura sin distracción:

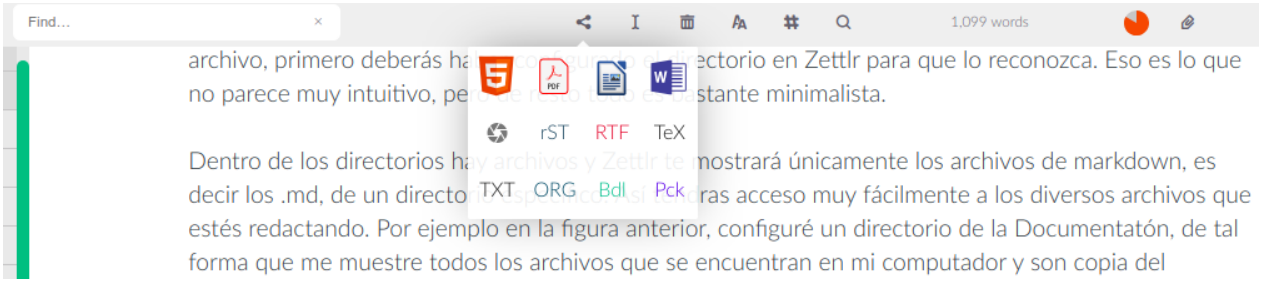
```
Ctrl + J
```

Así podrás ocultar la columna lateral izquierda y tener a pantalla completa el texto que escribes.

La exportación multiformato

Una vez configurado *Pandoc* y *Latex* en Zettlr, se pueden realizar exportaciones de los archivos de Markdown a múltiples formatos, como HTML 5, PDF, ODT e incluso a Reveal.js, para crear presentaciones interactivas desde archivos Markdown.

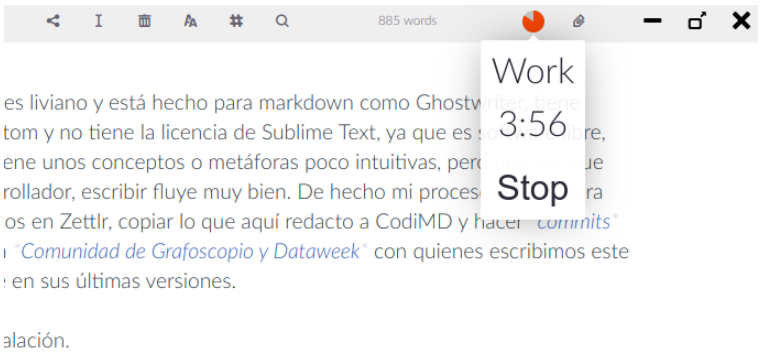
Si bien estas esportaciones se pueden realizar con *Pandoc*, resulta muy agradable tener en una interfaz gráfica estas opciones para realizarlas con un sólo click.



Captura de pantalla de la exportación a múltiples formatos

El Pomodoro-Timer o Cronómetro Pomodoro

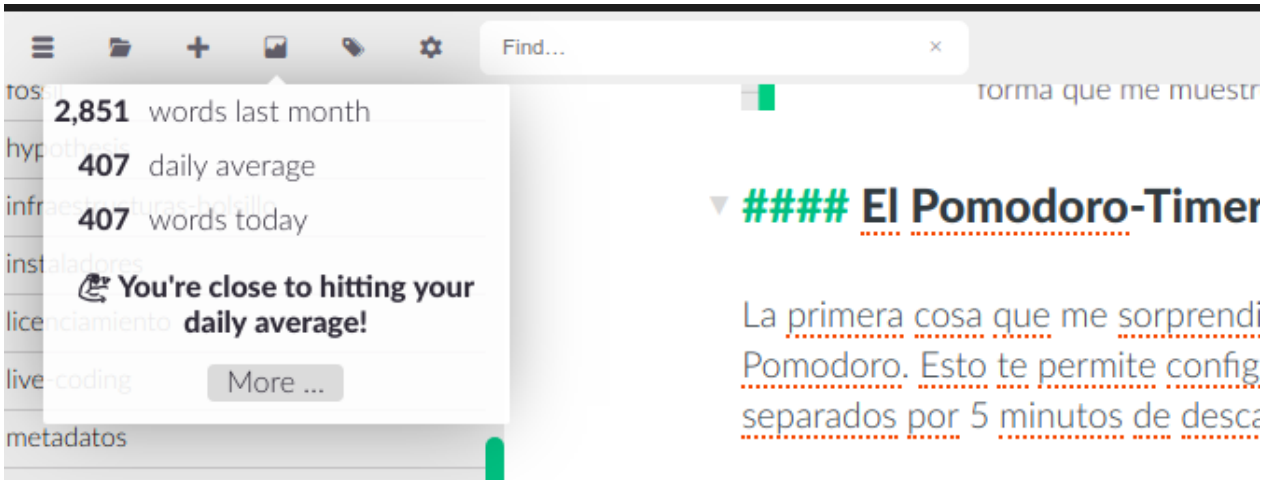
La primera cosa que me sorprendió, al igual que el *modo Hemingway* de Ghostwriter es el cronómetro Pomodoro. Esto te permite configurar lapsos de escritura, por ejemplo de 20 minutos cada uno, separados por 5 minutos de descanso.



Captura de pantalla del cronómetro Pomodoro en Zettlr

Las estadísticas

Zettlr tiene una sección de estadísticas, dode puedes ver cuánto escribes diariamente. Además podrás medir tu progreso o incluso seleccionar metas diarias.



Captura de pantalla de las estadísticas de Zettlr

> Que no pase un día sin leer o escribir por lo menos una línea
>
> Plinio el Viejo

Siguiendo esta premisa de Plinio el Viejo, Zettlr hace mediciones de cuánto escribes, para ayudarte a mantener un ritmo diario. Como puedes ver, Zettlr tiene muchos pequeños detalles para ayudarte a escribir, pero como dice el refrán, el Diablo está en los detalles.

Zettlr tiene muchas más opciones, pero para escribir Markdown fuera de línea puedes empezar con las aquí mencionadas. Igual te invitamos a que explores otros editores de Markdown para que encuentres lo que más te gusta e incluso si quieres recomendarnos otro, nos vendría muy bien saber de otras herramientas que podamos usar.

Fossil: Trabajo colaborativo distribuido

Imagen alusiva a Fossil

- **Propósito**: Introducir los sistemas de control de versiones, en especial, Fossil.
- **Ejercicio práctico**: Clonar un repositorio en Fossil, registrarse en él, sincronizarse y agregar nuevo contenido, hecho en Markdown.

- **Prerequisitos**:
  - **Lección**: Markdown.

Fossil es un sistema de colaboración distribuido para gestión de código fuente. Los SCM (Software Configuration Management) o DVCS (Distributed Version Control System) resuelven el problema respecto a cómo reproduzco el estado de un sistema de cómputo y su historia (orientado a archivos). Al resolver esta inquietud, se está resolviendo una pregunta incidental que es sobre cómo colaborar, de maneras no centralizadas.

Fossil es el SCM elegido para trabajar con Grafoscopio y el Data Week por su caracter sencillo y autocontenido, lo cual quiere decir que brinda mucha funcionalidad en un único programa, con flujos de trabajo y colaboración simples, y puede ejecutarse en una variedad grande de plataformas de hardware y software, con mínimos requerimientos de recursos de hardware (poca memoria RAM, menos de 3 Mb en disco duro) y software, además tanto los datos como los metadatos (salvo las contraseñas) son repartidos entre todos los usuarios que se sincronicen al repositorio, es decir, satisface nuestra definición de infraestructuras de bolsillo.

Existen algunas pocas herramientas para Git, como Gogs, que siguen el enfoque minimalista de Fossil, si bien no toda la información parece ser autocontenida e implican los flujos de trabajo complicados propios de Git.

Referencias Extra:

- Página principal de Fossil.
  - Son habituales las discusiones y comparaciones de Fossil contra los sistemas más populares, particularmente Git, así que acá algunos enlaces (de documentos en Inglés) que te ayudarán a tener una perspectiva informada, para defender el uso de Fossil vs Git, o para saber cuándo usar cuál:
    - Why You Should Use Fossil.
    - The Fossil Web Interface: Muestra varias de las características que Fossil integra desde su interfaz web. Veremos algunas de ellas en este mismo documento.
    - Preguntas y críticas.
    - Fossil vs Git.
    - Citas: Lo que la gente está diciendo.
    - Evaluaciones.
  - Documentación embebida para proyectos: Esta es la forma de documentación que empleamos para el Data Week y Grafoscopio. Explica cómo usarla y cómo acceder a la última versión de un

documento, o la que habrá de enviarse al repositorio, o la de alguna fecha o *commit* específicos.

- Reglas de Markdown usadas por Fossil.

- Fossil User Manual: Un texto detallado y con ejemplos sobre cómo usar Fossil en proyectos colaborativos. Si bien corresponde a una versión antigua de Fossil, las lógicas de uso y muchas de las interfaces siguen siendo utilizables hoy en día.

- Compilando Fossil con soporte para JSON.
- Clay Shirky: How someday Internet will one day transform the government: Una presentación, más bien simplista, sobre cómo estas tecnologías tipo Git y fossil podrían cambiar prácticas de poder, participación y democracía.
- Fossil en modelo bazar: Explica cómo usar Fossil en contextos donde no se conocen previamente los colaboradores.

  - Bazaar model with Fossil (hilo en el foro).
  - Bazar model with Fossil.
  - Hacking on the PiDP-8/I Software.
  - Contributing To Fossil.

Instalación

Antes de proceder a la instalación es conveniente verificar si ya se tiene instalado fossil (por ejemplo verificar si el comando «fossil» existe tecleando `$ which fossil`).

En Gnu/Linux

Averigua con el gestor de paquetes (apt, pacman) si tu distribución de Linux tiene Fossil disponible en una versión relativamente reciente (como las que usamos para nuestros talleres) e instálada desde dicho gestor.

En nuestra experiencia, algunas distribuciones derivadas de Debian, suelen tener paquetes muy viejos de Fossil.

Usando gestores/instaladores de software

Más información acá

Windows

Con Scoop

Importante: Para hacer estar parte debes haber instalado previamente Scoop.

Abrimos Power Shell y escribimos:

```
scoop install fossil
```

Al ejecutarlo, debemos ver algo como esto:

```
Installing 'fossil' (2.8) [32bit]
fossil-w32-2.8.zip (2.1 MB)
    [============================================================================================]
     100%
Checking hash of fossil-w32-2.8.zip ... ok.
Extracting fossil-w32-2.8.zip ... done.
Linking ~\scoop\apps\fossil\current => ~\scoop\apps\fossil\2.8
Creating shim for 'fossil'.
'fossil' (2.8) was installed successfully!
```

Métodos de instalación alternativos

Se puede instalar desde el gestor de paquetes en Linux, Mac, Windows, pero cuando está muy desactualizado, una alternativa es instalarlo desde el código fuente.

- Entramos a la página de descargas de Fossil y allí bajamos la última versión disponible para Windows.
- Una vez hallamos descompreso el archivo descargado, nos mostrará los contenidos del archivo zip, con un único elemento que es el archivo `fossil.exe`.
- Copiamos el archivo `fossil.exe` dentro del zip descargado y lo pegamos a la carpeta `C:Windows\`.
- Abrimos la consola de comandos en Windows y escribimos `fossil` en ella. Debe aparecer algo como:

Gnu/Linux y Mac

Así se hace esto en Gnu/Linux y Mac, usando comandos en la terminal:

1. Descargar el instalador desde la página web:

```
cd /tmp
wget http://fossil-scm.org/index.html/uv/fossil-linux-x64-2.3.tar.gz
```

2. Descomprimimos el archivo:

```
tar xvfz fossil-linux-x64-2.3.tar.gz
```

3. Encontraremos un binario llamado `fossil` que copiamos a donde están todos los binarios:

```
sudo cp fossil /usr/bin
```

Trabajar con repositorios

Para usar Fossil, vamos a sincronizarnos contra un repositorio, agregar archivos a este y mirar cómo ha cambiado su línea de tiempo. Hay otras cuestiones que vamos a dejar en el radar, pero que no vamos a profundizar, como el hecho de publicar repositorios propios. Sin embargo, ese tipo de funcionalidad también está provista por sistemas como Chisel.

Navegación

Existen determinados lugares para visitar con los cuales uno se puede hacer una idea de un repositorio de Fossil, sus contenidos y actividad. A continuación los listamos esos lugares, mostrando algunos ejemplos de los mismos en distintos repositorios.

- Portada:
  - Para Grafoscopio: http://mutabit.com/repos.fossil/grafoscopio/
  - Para el Data Week: http://mutabit.com/repos.fossil/dataweek
    (es la misma que está acortada en http://is.gd/oddbog)
- La línea de tiempo:

    - Data Week:
        - Por omisión: http://mutabit.com/repos.fossil/dataweek/timeline
        - Los últimos 500 «commits»: http://mutabit.com/repos.fossil/dataweek/timeline?n=500&y=all&t=&ms=exact

- Los contenidos (archivos o carpetas):
    - Para el Data Week: http://mutabit.com/repos.fossil/dataweek/dir

- Los «tickets» (solicitudes, peticiones, asuntos):
    - De Grafoscopio: http://mutabit.com/repos.fossil/grafoscopio/rptview?rn=1

Tickets: Sugerencias, correcciones y planeación

Una de las maneras más habituales de usar repositorios es a través de los *tickets* (boletas), que permiten realizar sugerencias sobre mejoras o ampliaciones, reportar errores para realizar correcciones e incluso hacer una gestión ligera de cómo el proyecto va dando cuenta de las solicitudes que se hacen sobre el mismo. Acá veremos cómo realizar cada una de estas actividades.

Crear un *ticket*

Empecemos por crear un *ticket*. Estos son los pasos:

- **Visitamos la sección de tickets**: en el del repositorio de Fossil respectivo cliqueamos en el enlace de *tickets*, usualmente referenciado en la portada o agregamos agregamos `/ticket` a la dirección principal del repositorio, por ejemplo para el repositorio de esta obra, la página de tickets es: https://mutabit.com/repos.fossil/documentaton/ticket. Veremos una imagen como la siguiente:
- **Revisamos los *tickets* preexistentes**: Esto lo hacemos antes de crear un nuevo ticket, de modo que podamos verificar que el nuestro no se encuentra ya reportado o podamos ver aquellos que estarían relacionados con el nuestro. Para esto cliqueamos el enlace que dice «All Tickets» y se nos mostrará una página como esta:

  La tabla de resumen previa nos muestra en la primera columna un identificador único para cada ticket, en la siguiente la fecha de creación, luego su tipo, el subsistema en el cual se presenta (en caso de ser reportado) y un título que resume el asunto del que se trata.

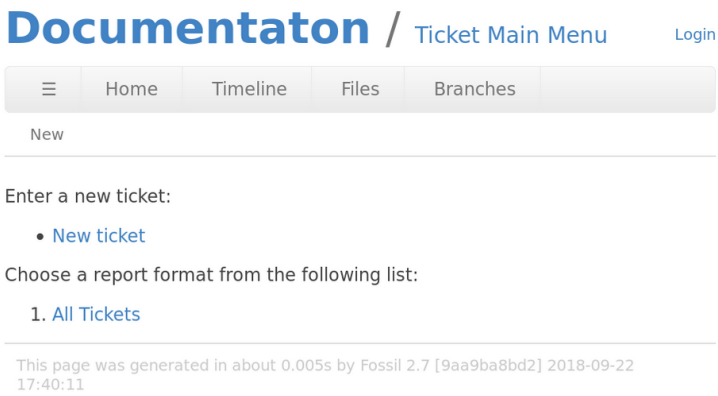


Figura 0.1: Página de *tickets* para un repositorio de Fossil (el de este libro)

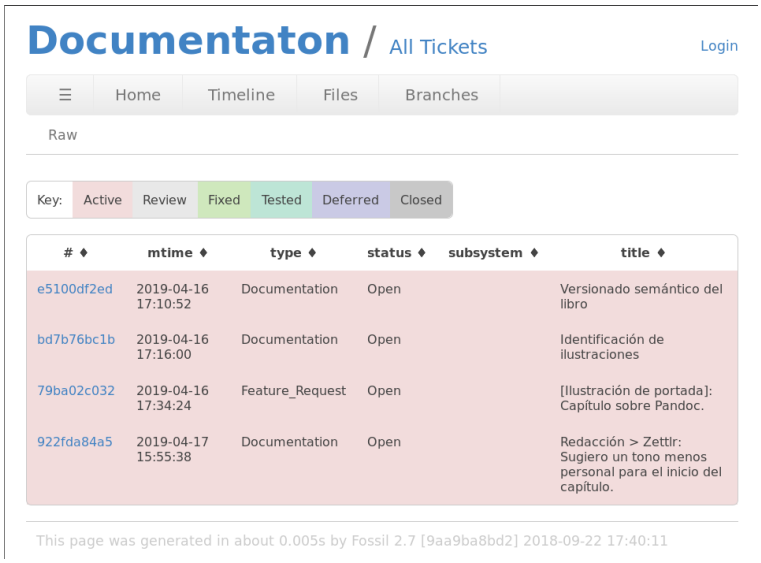
- **Miramos los detalles de un *ticket*** (Opcional) Al cliquear en el identificador único, en la tabla anterior, podemos ver información en mayor detalle del cada *ticket*, por ejemplo las conversaciones que se están teniendo al respecto y quiénes participan de las mismas. Esto es lo que vemos si hacemos click en el enlace e5100df2ed:

- **Creamos uno nuevo *ticket***: Esto lo hacemos en caso de que ningún *ticket* se refiera a aquello que queremos reportar, (de lo contrario es mejor si participamos en la conversación que ya existente al respecto). Para ello, hacemos click en «New Ticket», bien sea en la página de los detalles de cualquier ticket (ver figura 0.2) o en la página inicial (ver figura 0.1). Si no estamos acreditados en el repositorio veremos una página como la de la figura , que nos permitirá crear *tickets* de manera anónima, al tiempo que controla los mensajes indeseados en el repositorio (también conocidos como SPAM).

  Bien sea como usuarios anónimos o registrados en el repositorio, una vez cliqueamos el enlace de «New Ticket», veremos un formulario como este:

- **Llenamos el formulario**: eligimos las opciones del formulario que mejor describen lo que queremos y diligenciamos sus campos. A modo de ejemplo y siguiendo nuestra predilección por lo auto-referencial, crearemos un ticket sobre explicar la creación de nuevos tickets. El nombre de cada campo estará en *cursiva* y el valor diligenciado o seleccionado estará al frente en fuente `monoespaciada`. Así diligenciaremos los campos:

  - *Enter a one-line summary of the ticket*: `Explicar cómo crear nuevos tickets`.
  - *Type*: `Feature_Request`, pues estamos solicitando una nueva característica en el documento. Debido a los usos habituales de Fossil, la mayoría de las opciones para este campo están pensa-



Listado de todos los *tickets* de un reposiotrio

das para proyectos de software. `Code_Defect` se usa cuando estamos reportando errores en el software; `Build_Problem` para cuando es imposible crear el archivo derivado a partir de las fuentes (un binario, en caso de software o un PDF, EPUB o similares en el caso de documentos); `Documentation` se usa para la documentación que acompaña a los proyectos de software (pero en nuestro caso sería muy redudante, pues todo el proyecto es un documento).

- *Severity*: `Important`. Acá juzgamos qué tan clave es que dicho asunto sea resuelto dependiendo de cómo nos afecta a nosotros y al proyecto.
- *EMail:* Colocamos el correo si queremos ser contactados sobre la resolución del mismo. Obsérvese que también podemos seguir un proyecto a través de la suscripción a su archivo de sindicación en línea o RSS. (El del repositorio de este libro está en la portada).
- *Format*: `Markdown`. Es el formato en el cual escribiremos el reporte. La opción recomendad es Markdown, por todos los argumentos que ya hemos dado. Es importante anotar que Fossil tiene una variante especial, de modo que aquellas palabras entre paréntesis angulares (`[]`), se convierten en enlaces al repositorio, lo cual será útil para referenciar *commits* y eventos particulares, como veremos en la siguiente subsección.

- **Revisamos el formulario, lo re-editamos y enviamos**: Una vez diligenciado el formulario, veremos una vista previa muy similar a la del formulario previo, pero el código Markdown será previsualizado en esta y nos ofrecerá un botón de submit. Si necesitamos editar algo más, lo continuamos haciendo

y cliqueamos el botón `Preview`, de lo contrario hacemos click en el botón `Submit`, lo cual nos mostrará el ticket como aparece en el repositorio, como se muestra en la siguiente imagen.

Uno de los aspectos más importantes a tener en cuenta será el *Ticket UUID*, que es un identificador alfanumérico único del *ticket*, que usaremos para ubicarlo y referirnos al mismo luego. La primera parte de éste es el que aparece en la tabla donde se pueden ver todos los *tickets*, a la que ya nos hemos referido.

- Si hacemos click en la línea de tiempo y seleccionamos la opción *tickets* de la misma, veremos nuestro nuevo *ticket* como primero en esa lista, como se muestra a continuación:

  El la combinación alfanumérica que aparece entre paréntesis angulares es la versión resumida del *ticket UUID* y podemos usarla para enlazarlo desde otros lugares de Fossil, en particular desde otros *commits*. Este será el tema de la seguiente sección.

Cerrar un *ticket*

La sección precedente nos enseñó como abrir un *ticket* y nos dijo que esta era una de las formas más habituales de trabajo con los repositorios, como usuarios del proyecto que allí se construye. Esta sección muestra otra de las formas más habituales de trabajo, como co-autores del mismo: cerrar *tickets*.

Para esta sección, supondremos que tienes familiaridad con el resto del capítulo sobre Fossil. Si estás leyendo el libro en orden y sólo has leído hasta acá, es conveniente que la saltes y aprendas el resto de los elementos del trabajo básico con repositorios (clonación, modificación, etc), antes de proceder a cerrar el *ticket*. Hemos dejado esta sección acá, simplemente para manetener consistente la presentación.

Cerrar un *ticket* puede ocurrir por distintos motivos, pero el más usual de ellos es que hicimos trabajo dentro del repositorio para abordarlo. De este modo, los *commits* en el repositorio y los *tickets* pueden enlazarse, de modo que el trabajo hecho esté en el repositorio conectado con el reporte que solicitaba hacerlo.

Tomemos por ejemplo el ticket de la sección precedente. Una vez hemos empezado a explicar en la obra el aspecto que el *ticket* nos solicitaba, escribiendo y modificando los textos que dan cuenta de ello (bien sea en prosa o código) podemos hacer un commit indicándolo, con un comando como este:

```
1 fossil commit -m "Pandoc: Abordando ticket [921caf54a2]."
```

Veamos cómo queda la línea de tiempo en este caso:

Vemos quel el último commit dice algo como:

```
Pandoc: Abordando ticket [921caf54a2] Leaf check-in 9f8e9e98f1
```

es decir que hemos vinculado el check-in `9f8e9e98f1` con el ticket `921caf54a2` y si miramos los detalles del primero, veremos cuáles archivos cambiaron para poder abordar esta solicitud, en particular veremos esto:

Es decir, que un lector del repositorio está en condiciones de ver los archivos que se modificaron y en qué partes, para dar cuenta de determinada solicitud. Esto puede ocurrir para trozos de texto sustanciales, como en este caso, o porque hicimos cambios pequeños, como agrega una imagen a un capítulo o sus cabeceras, redimensionarla, o porque hicimos varias correciones de estilo u ortográficas. De este modo, cambios grandes o pequeños, solicitados por los lectores y usuarios de nuestra creación, pueden ser auditados asociándolos con los cambios efecticamente hechos en los archivos, para dar cuenta de los mismos.

Ahora bien, los vínculos son mejores en la web, si funcionan de doble vía. Cuando vayamos a cerrar el ticket, entrando en el mismo desde la tabla de resumen de todos ellos, mostrada en la sección precedente, es conveniente indicar que identificadores de los commits corresponde a aquellos donde se hizo trabajo en el *ticket*. Por ejemplo, ahora que se ha explicado un poco más sobre como cerrar el *ticket*, haremos otro *commit*, similar al anterior:

```
fossil commit -m "Fossil > Tickets: Cómo cerrarlos (ver [921caf54a2])."
```

Lo cual produce esta línea de tiempo ahora:

y cuando editemos el ticket `921caf54a2`, para cerrarlo. El formulario para cerrar ticket quedaría algo así:

Nótese que hemos cambiado el campo *Status* por `Closed` y el campo *Resolution* por `Fixed`, y en la descripción detalllada de cómo lo cerramos hacemos referencia a aquellos commits que hablaben del mismo en la línea de tiempo, en este ejemplo aquellos con los identifadores `[9f8e9e98f1]` y `[49406fac71]` (recordemos que al usar los paréntesis cuadrados, los convertimos automáticament en un enlace dentro del repositorio). Veamos cómo queda ahora la línea de tiempo:

El ticket que hemos cerrado aparece tachado (~~[921caf54a2]~~) y todos los estados de la línea de tiempo correspondientes a commits donde se referencio dicho ticket continuan enlazados al mismo. A su vez, si hacemos click en el ticket, veremos que apunta a los commits en líneas de tiempo donde trabajamos en el mismo:

De este modo Fossil nos provee una manera ligera de gestionar proyectos, mediante una práctica sencilla: enlazar *commits* y *tickets* entre sí. Una vez hallamos creado un ticket, usamos su UUID resumido y entre

paréntesis angulares para los commits que trabajen en el mismo y, una vez cerremos el ticket, hacemos lo propio para los commits, creando así una vinculación en doble vía. Si por alguna razón se reabre un *ticket*, es posible repetir este procedimiento con futuros *commits* para abordarlo y ediciones del mismo para cerrarlo.

Ya sea que queremos reportar y corregir errores o solicitar e implementar nuevas características en nuestra obra, esta manera de proceder nos permitirá indicar con agilidad qué hemos hecho en cualquiera de esos frentes y hacerlo transparente a la comunidad que se reune en torno a la obra.

Clonación

ADVERTENCIA: asegurarse que los comandos se ejecutan en el directorio correcto porque se pueden crear un montón de directorios y cosas en un directorio que no se desea.

Cuando se descarga un repositorio lo que se hace es traer toda la historia del mismo de su lugar remoto a mi disco duro. Vamos a hacer esto con el repositorio de la Documentanton, desde la consola de comandos.

- Creamos una carpeta para el repositorio del proyecto actual. Para Mac:

  ```
  mkdir ~/Documentos/<nombre_repositorio>
  ```

- Para Windows, ingresar a la linea de comandos (cmd):

  ```
  md ~/Documentos/<nombre_repositorio>
  ```

Donde <*nombre_repositorio*> es el nombre de la carpeta a descargar (repositorio).

- Nos ubicamos en esa carpeta creada:

  ```
  cd ~/Documentos/<nombre_repositorio>
  ```

- Clonamos el repositorio:

  ```
  fossil clone http://mutabit.com/repos.fossil/<nombre_repositorio>/ <
  nombre_repositorio>.fossil
  ```

  Al final de proceso deberá aparecer algo similar a:

  ```
  1     project-id: aa9f6e73b4f326e190f15c4a830c0909619bfa42
  2     server-id:  1f91261cb649d68c08dc75f7609975b787bc754f
  3     admin-user: Pibi (password is "acead4")
  ```

- Abrimos el repositorio:

  `fossil open <nombre_repositorio>.fossil`

  y se deben ver los mismos contenidos del repositorio remoto en nuestro disco duro local. Debera aparecer algo similar a:

```
project-name: Documentaton
repository:   D:/documentaton/documentaton.fossil
local-root:   D:/documentaton/
config-db:    C:/Users/palza/AppData/Local/_fossil
project-code: ac6959a5354c5b489888aa795e38efab1fdac313
checkout:     696a19768f970fb89a660e0ec6ffcbf5219eb205 2019-03-10
    00:23:10 UTC
tags:         trunk
comment:      initial empty check-in (user: offray)
check-ins:    1
```

Modificación

Vamos a crear un archivo en una subcarpeta dentro de la carpeta de idiomas, que contiene un de los capítulos del librillo.

- Crear la carpeta propia:

  `mkdir -p es/capitulos`

- Descargamos un archivo de Markdown (digamos la presentación) y lo salvamos con un nombre corto (digamos `presentacion.md`) dentro de esa carpeta.

- Agregamos ese archivo a ese repositorio

```
fossil add es/capitulos/presentacion.md
```

  Nota: Recuerda que Los símbolos < y > no se incluyen

  La respuesta debería ser algo como

```
ADDED es/capitulos/presentacion.md
```

- **Realizamos una actualización al repositorio** (en caso de que hayan habido cambios por otros mientras realizábamos los nuestros) y hacemos nuestro primer commit, con un mensaje descriptivo:

```
fossil update
fossil commit -m "<comentario corto descriptivo>."
```

Puede que recibamos mensajes sobre que no podemos autosincronizar. Por lo pronto no haremos caso de ellos.

Para ver lo que ocurre en el repositorio escribimos `fossil ui`, que despliega la interfaz gráfica de usuario en nuestra máquina. (Esto ocurre en menos de 3 Mb!). Debemos ver algo cómo:

Sincronización

La sincronización entre repositorios es hecha a través de un servidor que sirve como coordinador. En nuestro caso, será el repositorio original del Data Week. Lo anterior quiere decir que debemos registrarnos en dicho repositorio y solicitar permisos para sincronizarnos con él.

- Entramos a la página de ingreso (login), específicamente a la página de registro. Veremos algo como esto:

- Configurar usuario de fossil Esto se debe hacer si el usuario del sistema local (del pc) no es el mismo que se creó en el login del paso anterior. Fossil por defecto toma el nombre de usuario del sistema local (del PC).

  - Crear usuario nuevo en fossil local

```
fossil user new <usuario>
```

  - Ponerlo como usuario por defecto

```
fossil user default <usuario>
```

Donde es el usuario que creamos en el paso anterior del «login» (sin los símbolos < >).

- Sincronización: dentro del repositorio hacemos: Hacer un update primero (pues probablemente hay cambios nuevos que no se tienen): fossil update

```
fossil sync http://<usuario>@mutabit.com/repos.fossil/<nombre_repositorio>/
```

donde es el nombre del usuario con el que nos registramos en Fossil y es el nombre del repositorio (carpeta descargada)

En caso de que la actualización falle, es probable que hayas realizado una sincronización con un usuario incorrecto, verifica el usuario con que estas sincronizando, recuerda que fossil es sensible a minusculas y mayusculas.

Deberas tener una respuesta como esta:

```
Round-trips: 2   Artifacts sent: 0  received: 4
Sync done, sent: 1123  received: 3782  ip: 45.55.37.99
```

Cómo resolver las bifurcaciones involutarias (forks).

La diferencia entre «branches» y «forks»:

- What is the difference between a «branch» and a «fork»?.
- Branching, Forking, Merging, and Tagging Login.

El estado del repositorio a hoy era este:

Vemos que se han integrado dos bifurcaciones, pero aún quedan tres por integrar. Cada estado del sistema está descrito por un código alfanumérico único, suma hash.

Las ramas que fueron integradas corresponden (en la imagen) a las siguientes sumas hash: `[f0ea5717e4]` (El commit de Grace) y el tronco (trunk) de ese momento, cuyo *checksum* corresponde a `[1b5023d5a4]`, produciendo un estado del sistema nuevo, cuyo *checksum* es `[dcd4e53045]`.

Vamos a integrar una nueva ramificación en el tronco principal. Para ello, ubicados en el repositorio, desde la terminal

```
fossil update trunk
fossil merge 0ef40cd985
```

Aparecerá algo como esto:

```
Autosync:  http://offray@mutabit.com/repos.fossil/<nombre_repositorio>/
Round-trips: 1   Artifacts sent: 0  received: 0
Pull done, sent: 383  received: 2429  ip: 45.55.37.99
ADDED Participantes/Ivan/intro.md
```

```
 "fossil undo" is available to undo changes to the working checkout.
WARNING: multiple open leaf check-ins on trunk:
  (1) 2018-02-24 18:59:24 [dcd4e53045] (current)
  (2) 2018-02-24 01:33:03 [5da6cdb6dc]
  (3) 2018-02-24 01:32:45 [0a25a7636d]
  (4) 2018-02-24 01:31:51 [0ef40cd985]
```

Luego hacemos un commit al repositorio:

```
fossil commit -m "Integrando cambios de Iván."
```

Y luego mezclamos ese commit con el tronco principal, que de ahora en adelante llamaremos simplemente «trunk». Veremos algo como esto:

NOTA: Si olvidas hacer el `fossil update trunk` antes de hacer el merge, puedes usar `fossil undo` para revertir el cambio.

Varios

- **¿Cómo saber la URL del repositorio?**

  Tiene que ser dada por el «proveedor» del repositorio.

- **¿Qué quiere decir cuando aparece «Fossil internal error: repository doesn't not exist ... mapeda.fosil»?**

  Que se está intentando abrir un repositorio sin estar ubicado en la carpeta donde se clonó, o que se está intentando clonar en la carpeta de otro repositorio, sin haberlo cerrado previamente

- Ayuda en fossil: `fossil help COMMAND` (con `fossil help` da la lista de posibles comandos)

Figura 0.2: Detalles para un *ticket*.

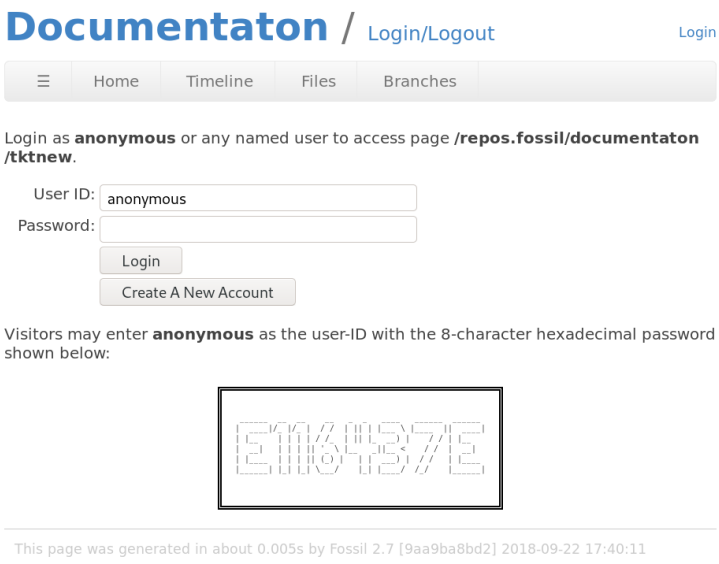
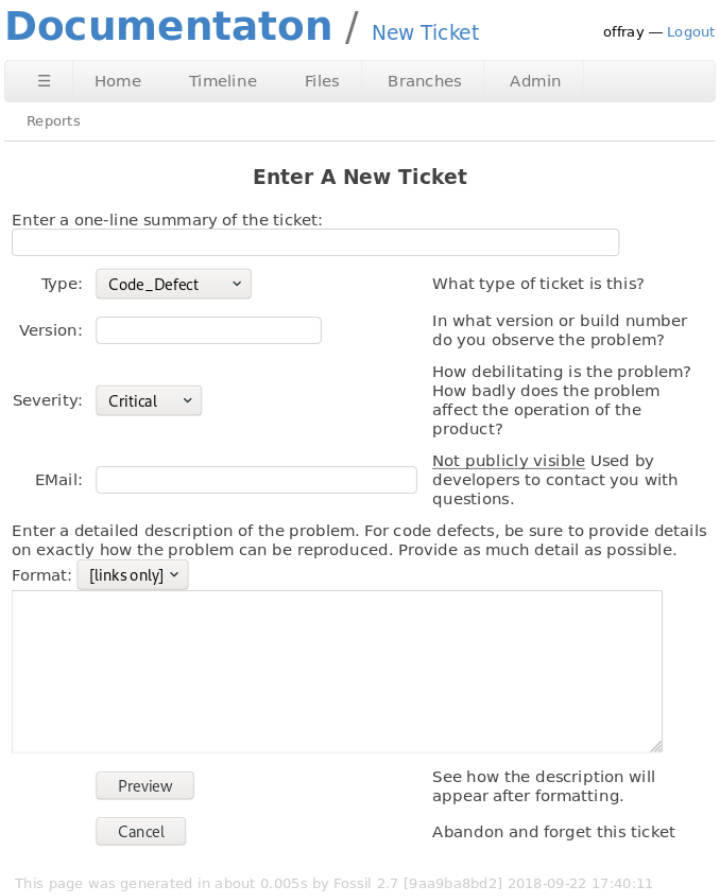
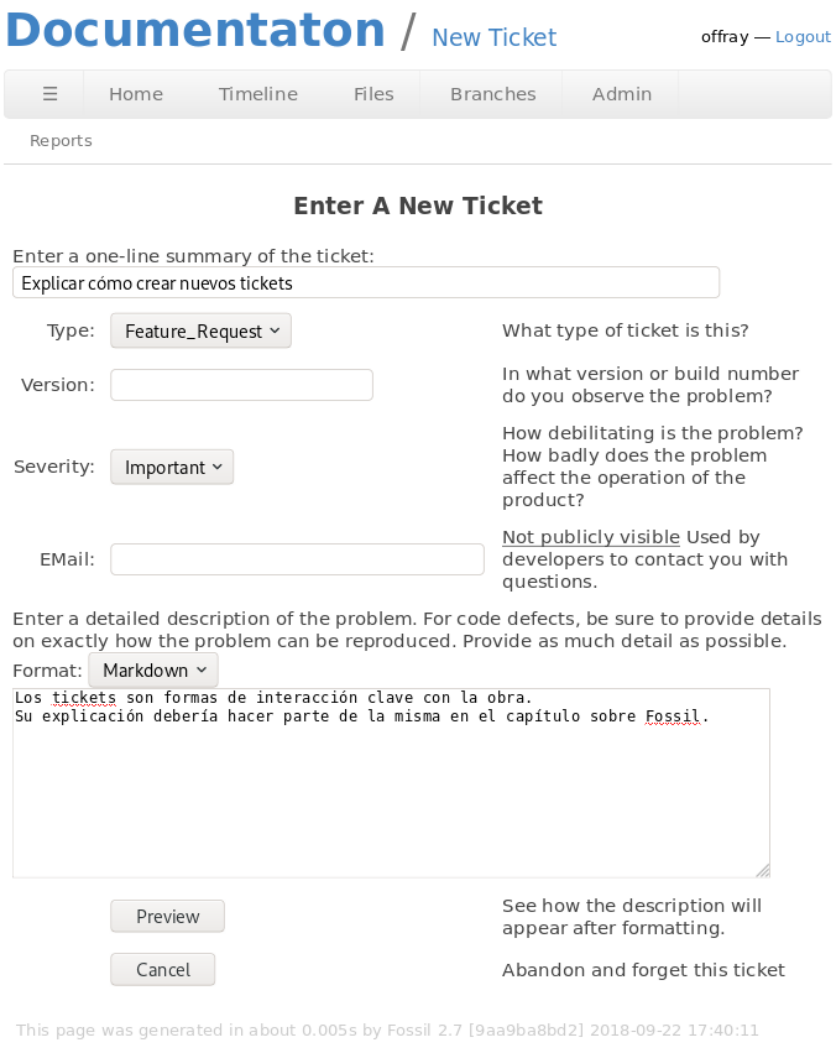


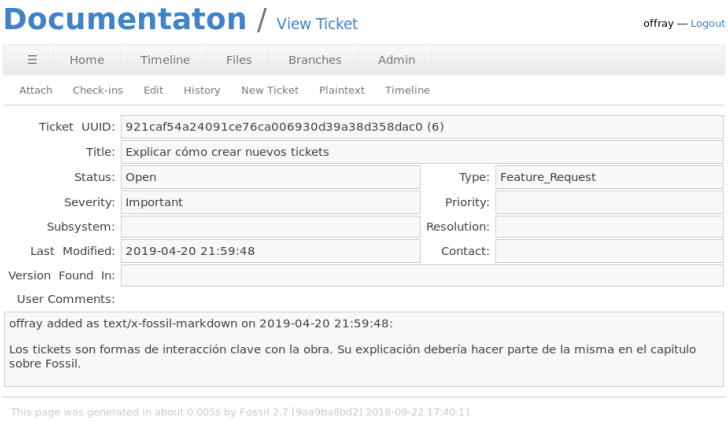
Figura 0.3: Ingreso para usuarios anónimos reportando *tickets*.



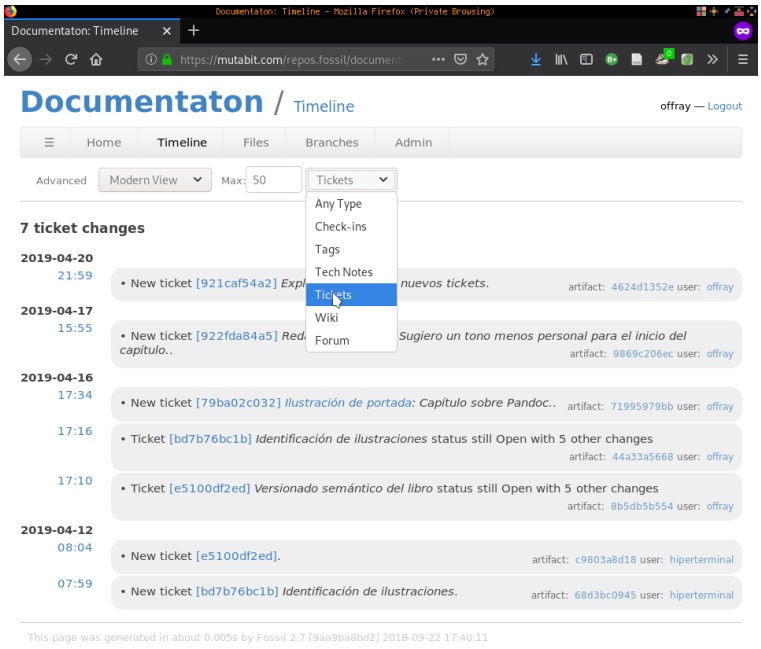
Formulario para la creación de un nuevo *ticket*.



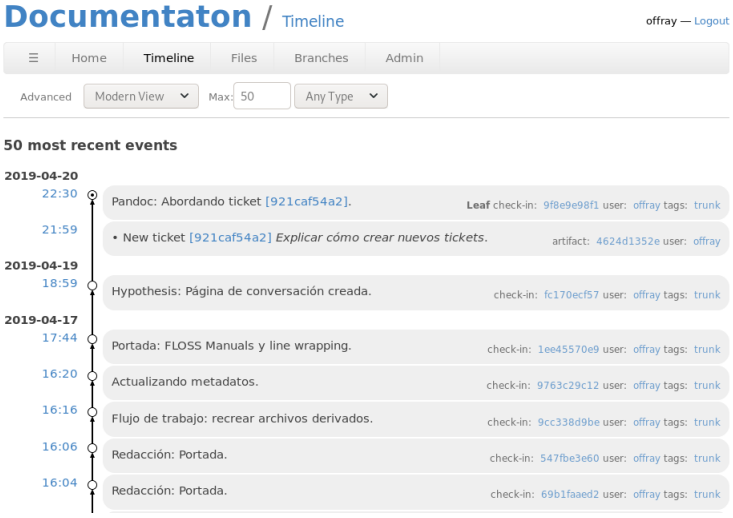
Formulario de nuevo *ticket* diligenciado.



Vista del *ticket* diligenciado.



Linea de tiempo para los *tickets*.



Línea de tiempo donde el *commit* se refiere al *ticket*.

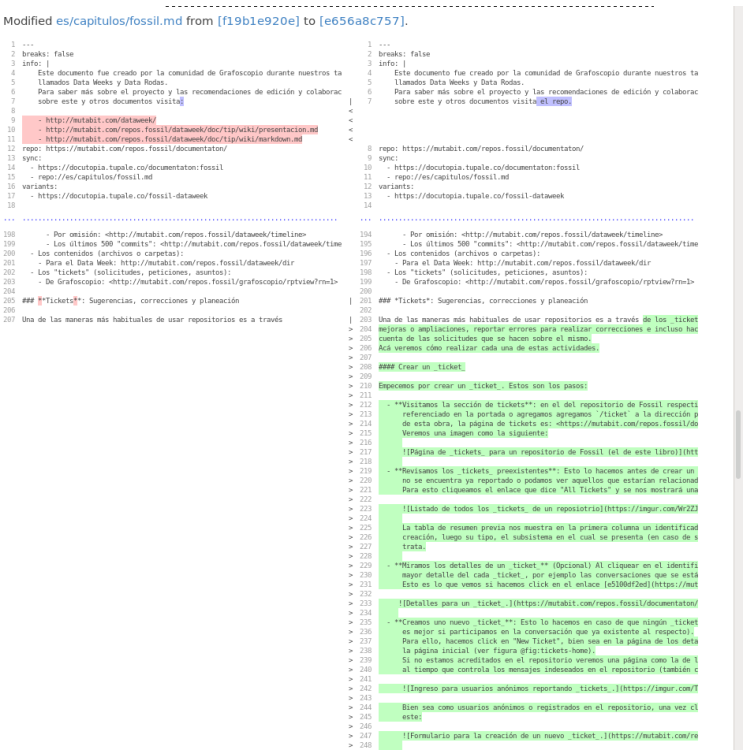
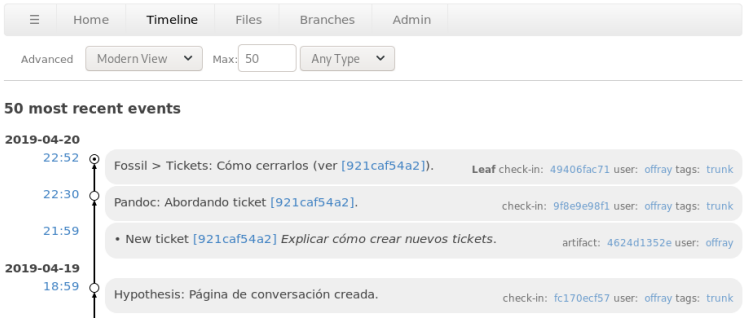
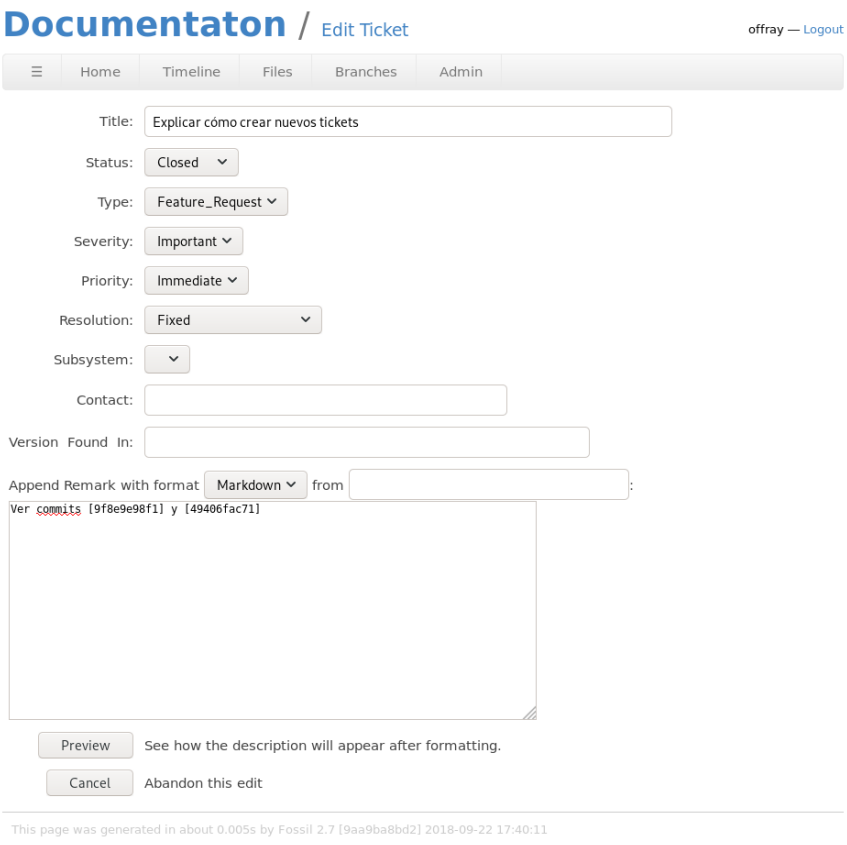


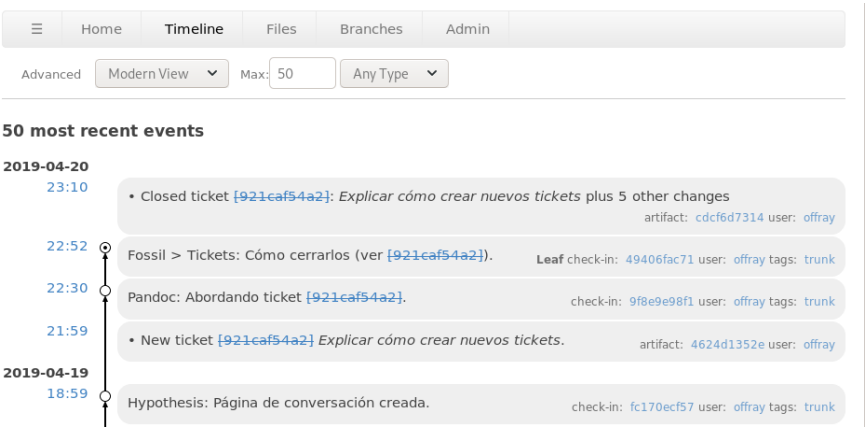
Figura 0.4: Diff en el archivo sobre Fosil: muestra lo que se agregó (en verde) y removió (en rojo) de un archivo mientras se explicaba como funcionan los *tickets*



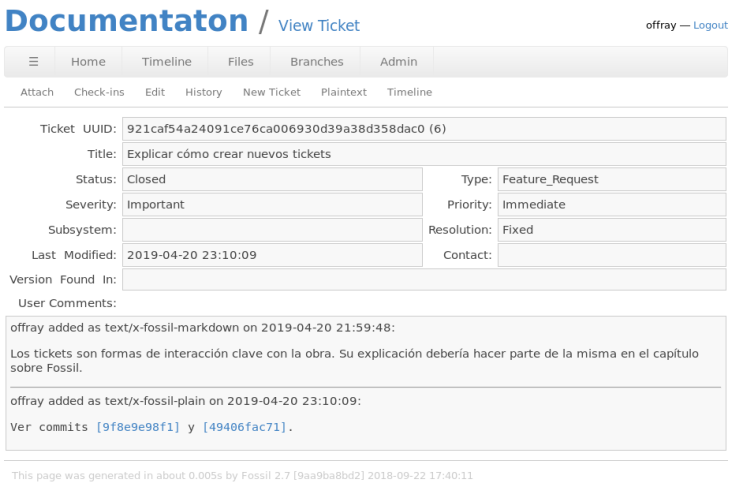
Segundo commit para cerrar un ticket



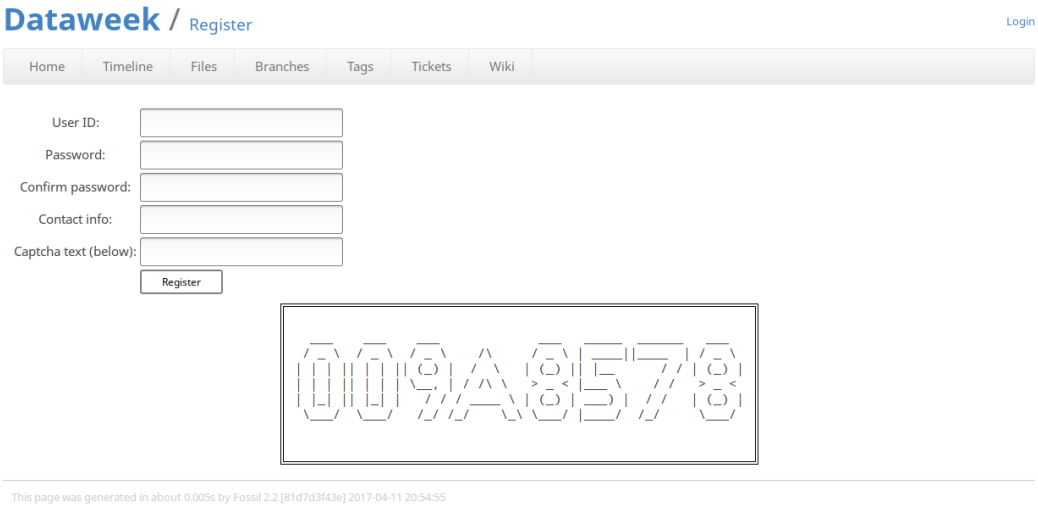
Editando el ticket para cerrarlo.



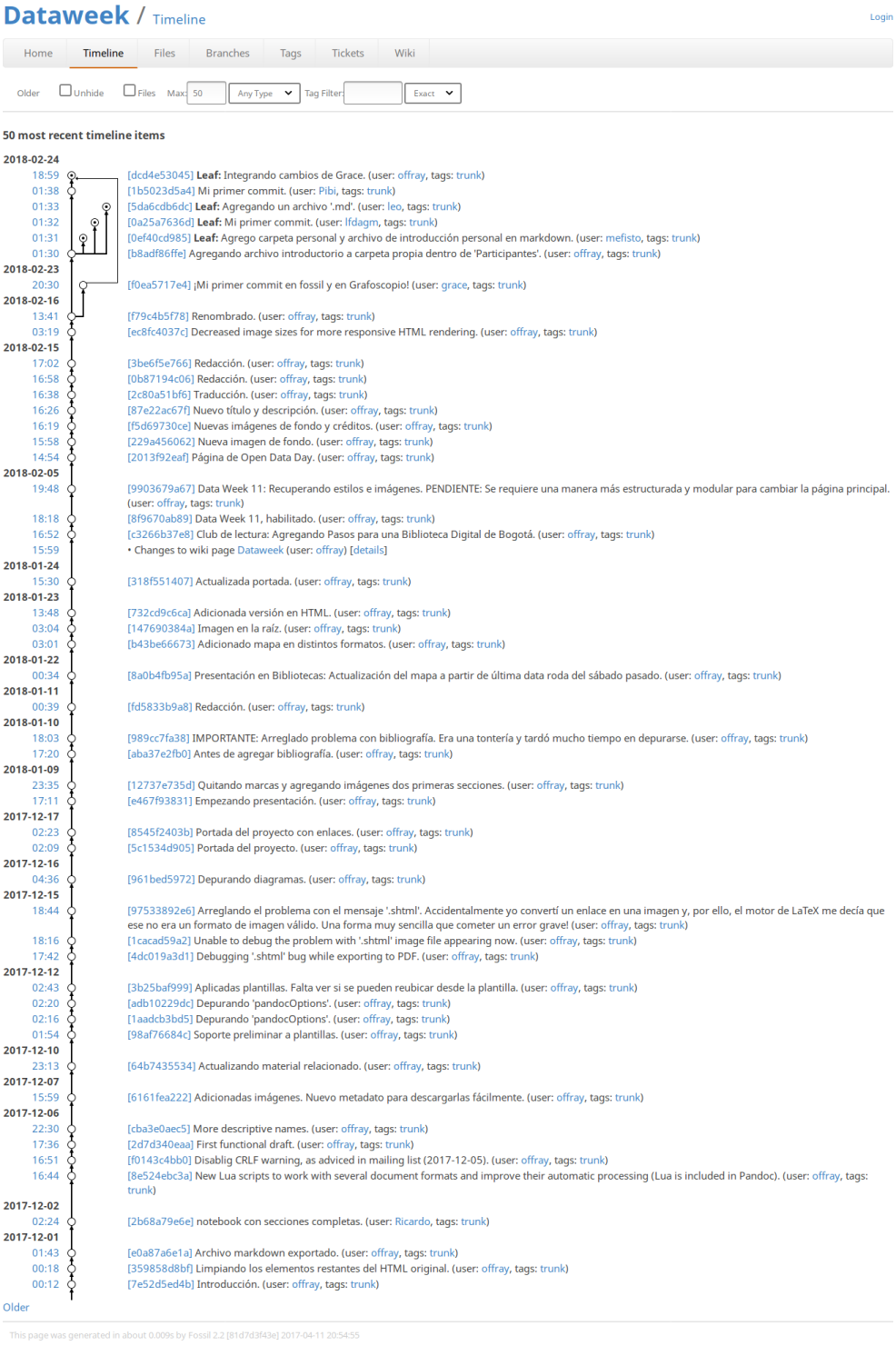
Línea de tiempo una vez se ha cerrado el *ticket*.



Detalle de *ticket* cerrado



Pantalla de registro de Fossil



Dataweek / Timeline

Login

Home | Timeline | Files | Branches | Tags | Tickets | Wiki

Older | Unhide | Files | Max: 50 | Any Type | Tag Filter: | Exact

50 most recent timeline items

2018-02-24

18:59 [dcd4e53045] **Leaf:** Integrando cambios de Grace. (user: offray, tags: trunk)
01:38 [1b5023d5a4] Mi primer commit. (user: Pibi, tags: trunk)
01:33 [5da6cdb6dc] **Leaf:** Agregando un archivo '.md'. (user: leo, tags: trunk)
01:32 [0a25a7636d] **Leaf:** Mi primer commit. (user: lfdagm, tags: trunk)
01:31 [0ef40cd985] **Leaf:** Agrego carpeta personal y archivo de introducción personal en markdown. (user: mefisto, tags: trunk)
01:30 [b8adf86ffe] Agregando archivo introductorio a carpeta propia dentro de 'Participantes'. (user: offray, tags: trunk)

2018-02-23

20:30 [f0ea5717e4] ¡Mi primer commit en fossil y en Grafoscopio! (user: grace, tags: trunk)

2018-02-16

13:41 [f79c4b5f78] Renombrado. (user: offray, tags: trunk)
03:19 [ec8fc4037c] Decreased image sizes for more responsive HTML rendering. (user: offray, tags: trunk)

2018-02-15

17:02 [3be6f5e766] Redacción. (user: offray, tags: trunk)
16:58 [0b87194c06] Redacción. (user: offray, tags: trunk)
16:38 [2c80a51bf6] Traducción. (user: offray, tags: trunk)
16:26 [87e22ac67f] Nuevo título y descripción. (user: offray, tags: trunk)
16:19 [f5d69730ce] Nuevas imágenes de fondo y créditos. (user: offray, tags: trunk)
15:58 [229a456062] Nueva imagen de fondo. (user: offray, tags: trunk)
14:54 [2013f92eaf] Página de Open Data Day. (user: offray, tags: trunk)

2018-02-05

19:48 [9903679a67] Data Week 11: Recuperando estilos e imágenes. PENDIENTE: Se requiere una manera más estructurada y modular para cambiar la página principal. (user: offray, tags: trunk)
18:18 [8f9670ab89] Data Week 11, habilitado. (user: offray, tags: trunk)
16:52 [c3266b37e8] Club de lectura: Agregando Pasos para una Biblioteca Digital de Bogotá. (user: offray, tags: trunk)
15:59 • Changes to wiki page Dataweek (user: offray) [details]

2018-01-24

15:30 [318f551407] Actualizada portada. (user: offray, tags: trunk)

2018-01-23

13:48 [732cd9c6ca] Adicionada versión en HTML. (user: offray, tags: trunk)
03:04 [147690384a] Imagen en la raíz. (user: offray, tags: trunk)
03:01 [b43be66673] Adicionado mapa en distintos formatos. (user: offray, tags: trunk)

2018-01-22

00:34 [8a0b4fb95a] Presentación en Bibliotecas: Actualización del mapa a partir de última data roda del sábado pasado. (user: offray, tags: trunk)

2018-01-11

00:39 [fd5833b9a8] Redacción. (user: offray, tags: trunk)

2018-01-10

18:03 [989cc7fa38] IMPORTANTE: Arreglado problema con bibliografía. Era una tontería y tardó mucho tiempo en depurarse. (user: offray, tags: trunk)
17:20 [aba37e2fb0] Antes de agregar bibliografía. (user: offray, tags: trunk)

2018-01-09

23:35 [12737e735d] Quitando marcas y agregando imágenes dos primeras secciones. (user: offray, tags: trunk)
17:11 [e467f93831] Empezando presentación. (user: offray, tags: trunk)

2017-12-17

02:23 [8545f2403b] Portada del proyecto con enlaces. (user: offray, tags: trunk)
02:09 [5c1534d905] Portada del proyecto. (user: offray, tags: trunk)

2017-12-16

04:36 [961bed5972] Depurando diagramas. (user: offray, tags: trunk)

2017-12-15

18:44 [97533892e6] Arreglando el problema con el mensaje '.shtml'. Accidentalmente yo convertí un enlace en una imagen y, por ello, el motor de LaTeX me decía que ese no era un formato de imagen válido. Una forma muy sencilla que cometer un error grave! (user: offray, tags: trunk)
18:16 [1cacad59a2] Unable to debug the problem with '.shtml' image file appearing now. (user: offray, tags: trunk)
17:42 [4dc019a3d1] Debugging '.shtml' bug while exporting to PDF. (user: offray, tags: trunk)

2017-12-12

02:43 [3b25baf999] Aplicadas plantillas. Falta ver si se pueden reubicar desde la plantilla. (user: offray, tags: trunk)
02:20 [adb10229dc] Depurando 'pandocOptions'. (user: offray, tags: trunk)
02:16 [1aadcb3bd5] Depurando 'pandocOptions'. (user: offray, tags: trunk)
01:54 [98af76684c] Soporte preliminar a plantillas. (user: offray, tags: trunk)

2017-12-10

23:13 [64b7435534] Actualizando material relacionado. (user: offray, tags: trunk)

2017-12-07

15:59 [6161fea222] Adicionadas imágenes. Nuevo metadato para descargarlas fácilmente. (user: offray, tags: trunk)

2017-12-06

22:30 [cba3e0aec5] More descriptive names. (user: offray, tags: trunk)
17:36 [2d7d340eaa] First functional draft. (user: offray, tags: trunk)
16:51 [f0143c4bb0] Disablig CRLF warning, as adviced in mailing list (2017-12-05). (user: offray, tags: trunk)
16:44 [8e524ebc3a] New Lua scripts to work with several document formats and improve their automatic processing (Lua is included in Pandoc). (user: offray, tags: trunk)

2017-12-02

02:24 [2b68a79e6e] notebook con secciones completas. (user: Ricardo, tags: trunk)

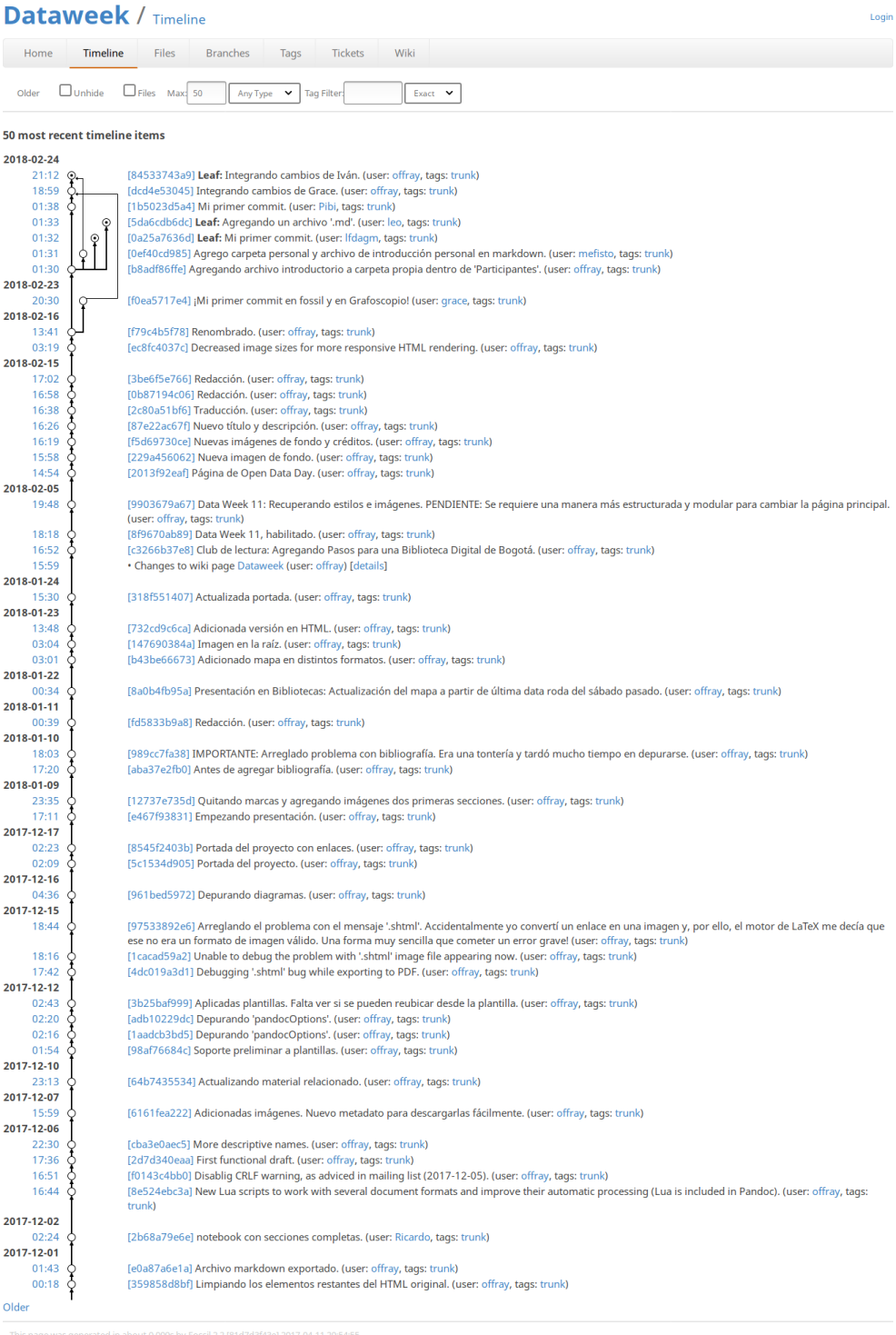
2017-12-01

01:43 [e0a87a6e1a] Archivo markdown exportado. (user: offray, tags: trunk)
00:18 [359858d8bf] Limpiando los elementos restantes del HTML original. (user: offray, tags: trunk)
00:12 [7e52d5ed4b] Introducción. (user: offray, tags: trunk)

Older

This page was generated in about 0.009s by Fossil 2.2 [81d7d3f43e] 2017-04-11 20:54:55

Repositorio antes del merge.



Dataweek / Timeline

Login

Home | Timeline | Files | Branches | Tags | Tickets | Wiki

Older ☐ Unhide ☐ Files Max: 50 Any Type Tag Filter: Exact

50 most recent timeline items

2018-02-24

21:12 [84533743a9] **Leaf:** Integrando cambios de Iván. (user: offray, tags: trunk)
18:59 [dcd4e53045] Integrando cambios de Grace. (user: offray, tags: trunk)
01:38 [1b5023d5a4] Mi primer commit. (user: Pibi, tags: trunk)
01:33 [5da6cdb6dc] **Leaf:** Agregando un archivo '.md'. (user: leo, tags: trunk)
01:32 [0a25a7636d] **Leaf:** Mi primer commit. (user: lfdagm, tags: trunk)
01:31 [0ef40cd985] Agrego carpeta personal y archivo de introducción personal en markdown. (user: mefisto, tags: trunk)
01:30 [b8adf86ffe] Agregando archivo introductorio a carpeta propia dentro de 'Participantes'. (user: offray, tags: trunk)

2018-02-23

20:30 [f0ea5717e4] ¡Mi primer commit en fossil y en Grafoscopio! (user: grace, tags: trunk)

2018-02-16

13:41 [f79c4b5f78] Renombrado. (user: offray, tags: trunk)
03:19 [ec8fc4037c] Decreased image sizes for more responsive HTML rendering. (user: offray, tags: trunk)

2018-02-15

17:02 [3be6f5e766] Redacción. (user: offray, tags: trunk)
16:58 [0b87194c06] Redacción. (user: offray, tags: trunk)
16:38 [2c80a51bf6] Traducción. (user: offray, tags: trunk)
16:26 [87e22ac67f] Nuevo título y descripción. (user: offray, tags: trunk)
16:19 [f5d69730ce] Nuevas imágenes de fondo y créditos. (user: offray, tags: trunk)
15:58 [229a456062] Nueva imagen de fondo. (user: offray, tags: trunk)
14:54 [2013f92eaf] Página de Open Data Day. (user: offray, tags: trunk)

2018-02-05

19:48 [9903679a67] Data Week 11: Recuperando estilos e imágenes. PENDIENTE: Se requiere una manera más estructurada y modular para cambiar la página principal. (user: offray, tags: trunk)
18:18 [8f9670ab89] Data Week 11, habilitado. (user: offray, tags: trunk)
16:52 [c3266b37e8] Club de lectura: Agregando Pasos para una Biblioteca Digital de Bogotá. (user: offray, tags: trunk)
15:59 • Changes to wiki page Dataweek (user: offray) [details]

2018-01-24

15:30 [318f551407] Actualizada portada. (user: offray, tags: trunk)

2018-01-23

13:48 [732cd9c6ca] Adicionada versión en HTML. (user: offray, tags: trunk)
03:04 [147690384a] Imagen en la raíz. (user: offray, tags: trunk)
03:01 [b43be66673] Adicionado mapa en distintos formatos. (user: offray, tags: trunk)

2018-01-22

00:34 [8a0b4fb95a] Presentación en Bibliotecas: Actualización del mapa a partir de última data roda del sábado pasado. (user: offray, tags: trunk)

2018-01-11

00:39 [fd5833b9a8] Redacción. (user: offray, tags: trunk)

2018-01-10

18:03 [989cc7fa38] IMPORTANTE: Arreglado problema con bibliografía. Era una tontería y tardó mucho tiempo en depurarse. (user: offray, tags: trunk)
17:20 [aba37e2fb0] Antes de agregar bibliografía. (user: offray, tags: trunk)

2018-01-09

23:35 [12737e735d] Quitando marcas y agregando imágenes dos primeras secciones. (user: offray, tags: trunk)
17:11 [e467f93831] Empezando presentación. (user: offray, tags: trunk)

2017-12-17

02:23 [8545f2403b] Portada del proyecto con enlaces. (user: offray, tags: trunk)
02:09 [5c1534d905] Portada del proyecto. (user: offray, tags: trunk)

2017-12-16

04:36 [961bed5972] Depurando diagramas. (user: offray, tags: trunk)

2017-12-15

18:44 [97533892e6] Arreglando el problema con el mensaje '.shtml'. Accidentalmente yo convertí un enlace en una imagen y, por ello, el motor de LaTeX me decía que ese no era un formato de imagen válido. Una forma muy sencilla que cometer un error grave! (user: offray, tags: trunk)
18:16 [1cacad59a2] Unable to debug the problem with '.shtml' image file appearing now. (user: offray, tags: trunk)
17:42 [4dc019a3d1] Debugging '.shtml' bug while exporting to PDF. (user: offray, tags: trunk)

2017-12-12

02:43 [3b25baf999] Aplicadas plantillas. Falta ver si se pueden reubicar desde la plantilla. (user: offray, tags: trunk)
02:20 [adb10229dc] Depurando 'pandocOptions'. (user: offray, tags: trunk)
02:16 [1aadcb3bd5] Depurando 'pandocOptions'. (user: offray, tags: trunk)
01:54 [98af76684c] Soporte preliminar a plantillas. (user: offray, tags: trunk)

2017-12-10

23:13 [64b7435534] Actualizando material relacionado. (user: offray, tags: trunk)

2017-12-07

15:59 [6161fea222] Adicionadas imágenes. Nuevo metadato para descargarlas fácilmente. (user: offray, tags: trunk)

2017-12-06

22:30 [cba3e0aec5] More descriptive names. (user: offray, tags: trunk)
17:36 [2d7d340eaa] First functional draft. (user: offray, tags: trunk)
16:51 [f0143c4bb0] Disablig CRLF warning, as adviced in mailing list (2017-12-05). (user: offray, tags: trunk)
16:44 [8e524ebc3a] New Lua scripts to work with several document formats and improve their automatic processing (Lua is included in Pandoc). (user: offray, tags: trunk)

2017-12-02

02:24 [2b68a79e6e] notebook con secciones completas. (user: Ricardo, tags: trunk)

2017-12-01

01:43 [e0a87a6e1a] Archivo markdown exportado. (user: offray, tags: trunk)
00:18 [359858d8bf] Limpiando los elementos restantes del HTML original. (user: offray, tags: trunk)

Older

This page was generated in about 0.009s by Fossil 2.2 [81d7d3f43e] 2017-04-11 20:54:55

Repositorio después del merge.

Pandoc: Exportación multiformato

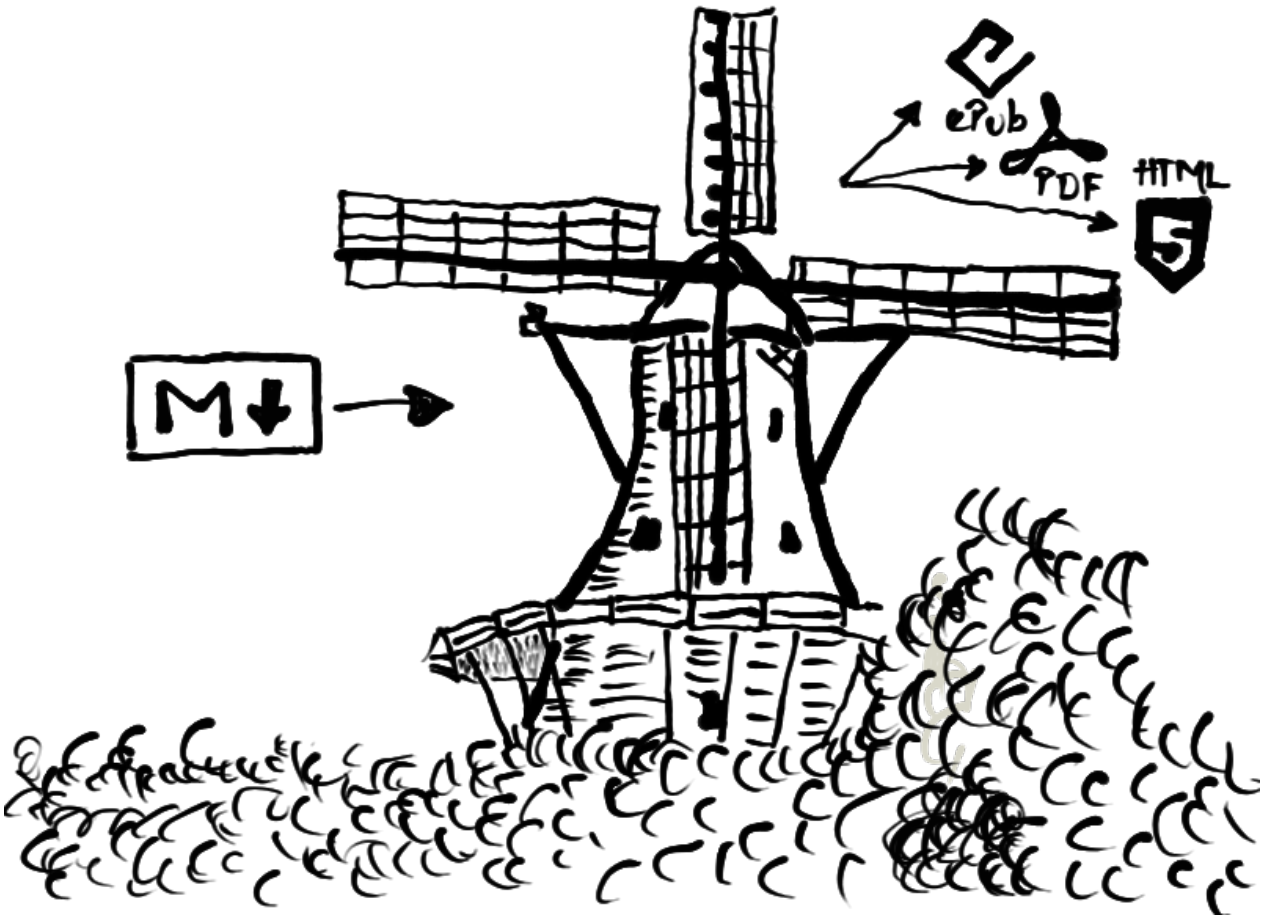


Imagen alusiva a la exportación multiformato de Pandoc

- **Objetivo**: Aprender a compilar nuestros escritos en una única obra y exportarla en múltiples formatos.

- **Prerrequisitos**:
  - Markdown
- **Recomendaciones**
  - CodiMD
  - Fossil.

Hasta ahora hemos aprendido cómo escribir en formatos ágiles de manera individual. Si has seguido la recomendaciones, en este punto sabrás también cómo extender esta escritura para que sea colectiva y como hacer que dicha escritura sea resiliente, almacenando copias que permiten guardar nuestros escritos y su historia de manera distribuida.

Ahora llega el momento de compilar esta obra en una secuencia ordenada y exportarla en distintos formatos.

Nuestra herramienta principal de exportación será Pandoc, que es una herramienta para transformar entre múltiples formatos de etiquetamiento ligero para representar texto estructurado, creada y liderada por John MacFarlane, que además de la manipulación entre formatos, nos permite compilarlos en una única obra creada a partir de documentos distintos.

Instalación

Linux y MacOS

Para esta parte debes tener instalado el gestor de paquetes Nix.

```
1    nix-env -i pandoc
```

O puedes usar los métodos de instalación de tu distribución de Gnu/Linux particular.

Windows

Para esta parte debes tener instalado Chocolatey.

Abre la consola y ejecuta:

```
choco install pandoc
```

También es conveniente instalar MiKTeX para la creación de PDFs:

```
choco install rsvg-convert python miktex
```

Al instalar MiKteX, la instrucción anterior te podrá solicitar confirmación para correr los scripts, presiona «Y» para avanzar en la instalación de paquetes intermedios requeridos.

Exportación de Markdown a PDF

Para exportar a distintos formatos lo que hacemos en los suguiente:

```
pandoc archivo1.md archivo2.md archivoN.md --output compilado.pdf
```

Donde cada uno de los archivos `.md` son archivos de Markdown que ordenados constituiran el PDF final.

En ocasiones quisiéramos cambiar también las opciones de fuentes, paginación, títulos, tipografías, tamaño de página, etc. Usualmente expresamos todas ellas en un archivo de metadatos, llamado, por convención, `metadata.yaml`. Si queremos usar las opciones en ese archivo, para controlar nuestro PDF lo que hacemos es lo siguiente:

```
pandoc metadata.yaml archivo1.md archivo2.md archivoN.md --output
    compilado.pdf
```

Por ejemplo, para el libro de Feminismo de Datos el archivo yaml de opciones es este.

Ejemplo: este librito/cartilla en PDF.

Corremos el comando dentro de la carpeta de español para producir la versión del libro:

```
cd es/
```

```
pandoc metadata.yaml \
    capitulos/presentacion.md  \
    capitulos/convenciones.md \
```

```
    capitulos/fossil.md  \
    capitulos/infraestructuras-bolsillo.md \
    capitulos/instaladores.md \
    -o documentaton.pdf
```

Veamos lo que hemos hecho en mayor detalle:

- Primero nos ubicamos en la carpeta del idioma donde están los textos que consitutyen nuestro libro.
- Luego empleamos Pandoc para exportarlo a distintos formatos, para ello indicamos cuál era el archivo de metadatos (`metadata.yaml`), que continene las opciones de personalización y luego cada uno de los capítulos en orden. En sistemas *nix el backslash (\) funciona dentro de archivos y órdenes ejecutables, indicando saltos de línas que van concatenadas y lo usamos para fácil lectura, colocando un argumento en una línea diferente.

Agregamos el archivo en PDF como archivo no versionado dentro del repositorio, es decir que no rastrearemos los cambios en el mismo.

Desde la raíz del repositorio hacemos:

```
fossil uv add es/documentaton.pdf
fossil uv sync -v
```

Edición y depuración de las fuentes y los archivos derivados

Este es en líneas generales el flujo de trabajo para participar de la co-creación de esta obra (en detalle se explican el en texto mismo):

1. Lee la página de portada, para que tengas claro la intensión general del trabajo y sus métodos, públicos y herramientas, así como las formas de contacto y ayuda.
2. Escribe el texto nuevo en o fuera de línea (CodiMD o Zettlr) y agrégalo al repositorio.
3. Actualiza el repositorio para sincronizarte con los otros cambios que hayan ocurrido.
4. Haz commit de tu nuevo texto al repositorio.
5. Si se trata de una edición a un texto que ya está en el repositorio, edítalo directamente en el repositorio y repite los pasos 3 y 4.
6. Actualiza la portada para reflejar los nuevos contenidos que has puesto.

7. Recrea los archivos derivados (PDF, EPUB, etc.) periódicamente y revisa que la incorporación de nuevo contenido mantenga las normas estilísticas y de forma del resultado (escalado de imágenes, metadatos propiamente adecuados, secciones y subsecciones, etc).
8. En caso de conflictos entre los pads (CodiMD) y el repositorio (Fossil), éste último se considera como la versión canónica del documento, pues es más fácil trazar los cambios para documentos en Fossil, que en CodiMD.

Filtros

Uno de los balances más interesantes que hace Pandoc es tener un lenguaje que sea sencillo, a la vez que sea extensible. Esto permite aprenderlo en poco tiempo y progresivamente, pues lo básico se maneja en pocos minutos y las caracterísiticas más avanzadas se adquieren con el tiempo. Dichas características incluyen el uso de macros de LaTeX para la escritura de matemática o el control más granular del texto y su presentación, así como el empleo de filtros, que permiten extender el lenguaje básico a la vez que mantienen su caracter neutral respecto a los distintos formatos de salida en que será representado. Nosotros hemos hecho uso de ambas características a lo largo de esta obra: LaTeX embebido y filtros. Haremos una introducción a la segunda en el contexto de esta y para mayores detalles se recomienda consultar la sección de Filtros del Manual de Pandoc.

Los filtros son programas externos a Pandoc que acceden al texto que éste procesa y transforma de un formato a otro y ayudan con dichas transformaciones, permitiendo así la personalización y extensión del lenguaje Markdown soportado por omisión por Pandoc. Pueden estar escritos en un gran número de lenguajes de *scripting* y existe soporte para varios populares como Python o Ruby, además Pandoc incluye ya a uno minimalista, llamado Lua.

Como verás, a lo largo de este manual hemos alternado de manera consistente explicaciones textuales con imágenes, figuras y capturas de pantalla. En ocasiones dicha alternacia sale bien, pero en otras las figuras terminan en otra página que no es continua al texto donde se mencionan o deben ser referidas de manera reiterada, por lo cual sería conveniente tener una manera de evocarlas y enlazarlas desde distintos lugares del texto. El filtro que vamos a usar nos permite, precisamente, referenciar figuras y se llama pandoc-fignos.

Para trabajar con `pandoc-fignos` supondremos que ya tenemos instalado Python y su manejador de paquetes PIP, pues está hecho en dicho lenguaje.

- Para instalarlo, desde la consola de comandos hacemos:

```
pip install pandoc-fignos
```

- Para actualizarlo a la versión más reciente:

```
pip install --upgrade pandoc-fignos
```

Para usarlo simplemente colocamos la referencia a la figura en los campos extra de Markdown precediso de un hashtag, algo como

```
    ![Mi figura](enlace/a/figura){#fig:identificador}
```

y para referenciar dicha figura, en cualquier parte del texto en Markdown decimos:

```
    Véase figura @fig:identificador.
```

Los identificadores de las figuras deben ser únicos y no confundirse con los de otras o alias de citas bibliográficas, pues usan el mismo caracter `@` para referirse tanto a figuras como a citas.

- ¿Puedes encontrar en este libro, lugares donde hayamos usado este filtro para referencias a figuras?
- ¿Encuentras algún uso de los filtros para los escritos que estás realizando?¿Cuáles?

Puede que hagamos un uso más detallado de los filtros, en posteriores capítulos (particularmente los de la parte 3) para automatizar y extender la conversión entre distintos formatos.

Cierre y continuaciones

Al cierre de esta parte deberías estar en condidiones de:

- ☐ Determinar cuáles son las difererencias de los lenguajes de etiquetamiento ligeros sobre otras formas de escritura más convecionales y decir por qué y en qué contextos son preferibles.
- ☐ Usar herramientas en línea y fuera de ella para escribir Markdown en solitario y colectivamente de manera síncrona y asíncrona.
- ☐ Compilar documentos escritos en Markdown en una obra única y exportarlos a diferentes formatos, en particular PDF y ePub.
- ☐ Guardar tus documentos en el gestor de código Fossil y sincronizar tu trabajo con el de otros a través del mismo.

La tercera parte indicará cómo automatizar este proceso de publicación, empleando para ello herramienta de programación amigables y ágiles.

Publicación Automatizada

En esta parte veremos como hacer publicación automatizada y nos introducirá a la programación y el *live coding* (nos adentraremos en estas definiciones luego) desde la perspectiva de las labores de publicación que llevamos haciendo hasta el momento.

Introducción y problema generador

Existen muchas maneras de llegar a la programación, y en la comunidad de Grafoscopio, desde donde escribimos este texto, hemos explorado varias, todas ellas vinculadas con alfabetismos críticos, es decir, aquellos que suponen que el analfabeta es aquel que carece de algunos conocimientos para participar plenamente de la transformación del mundo que quiere construir. En esta definición el alfabetismo no es un currículo impuesto desde afuera, por los que ya saben, sino una necesidad sentida desde las comunidades de base, desde los saberes que éstas consideran relevantes, en negociación con el mundo más amplio y sus maneras de participación y transformación de éste. Esto nos aleja de miradas más instrumentales frente a la creación de código y el trágico pero clásico ejemplo introductorio del «Hola Mundo!» y se acerca más a perspectivas críticas (como las de Paulo Freire). Por ello, suponemos que el código nos habilita para prácticas ciudadanas y del disfrute de mundo, y que podría ser un saber transversal, como la lectura o la escritura, para todos los ciudadanos y no un monopolio de los programadores, las manoseadas *startups* y la intensión no es *simplemente* hacer negocios, volverse multimillonario o crear un app para salvar el mundo. El contar con fronteras de saberes más amplios, nos ayuda a construir mejor un mundo compartido. Por eso necesitamos tanto programadores e ingenieros que sepan de estéticas y teorías críticas, como humanistas que sepan de código. El código se convierte en nuestra mirada en uno de esos saberes limítrofes y al alcance de muchos.

El código puede ser un importante lugar para la expresión y enacción del pensamiento crítico y la publicación es una de esas prácticas ciudadanas/lúdicas facilitadas por este, dentro de las cuales también contamos la recuperación y transparencia de datos del sector privado/público, las visualizaciones con fines estéticos o activistas, entre tantas otras. Como dijimos, varios de esos han sido nuestros accesos al mundo de la programación en la comunidad de Grafoscopio y acá miraremos uno en particular: la publicación.

Problemas generadores[1]

Nuestra aproximación a la escritura de código desde esta perspectiva está orientada por problemas y preguntas, desde las cuales vamos desarrollando nuestras prácticas educativas y de aprendizaje entre pares. Usualmente se trata de problemas abiertos, para los cuales nosotros mismos no conocemos la

[1] El problema generador es un término prestado de la pedagogía de Paulo Freire.

respuesta por completo y vamos adquiriendo los saberes que nos permiten respuestas parciales y en evolución.

El problema que orientará nuestro aprendizaje en este caso será:

¿Como hacer la publicación de un texto reproducible?, es decir, ¿cómo hacer que el texto que alguien obtuvo a partir de un conjunto de archivos, sea el mismo que obtenga otra persona?

Nuestra respuesta hasta el momento ha sido la de usar sistemas de control de versiones (en particular Fossil), para garantizar que esos archivos son los mismos en diferentes computadores y un comando para recrear el texto a partir de los archivos. La primera parte se enfrenta al almacenamiento de la información, mientras que la segunda tiene que ver con su procesamiento y transformación. Sin embargo, esta aproximación aún tiene algunos inconvenientes, particularmente frente a la segunda parte, el comando que usamos para compilar y exportar la obra. Veamos cuáles son:

- Podemos colocar el comando en un archivo de texto ejecutable, llamado *script*, que nos permita también versionarlo dentro de Fossil y esto nos daría gran compatibilidad, al menos entre sistemas tipo Unix (como Gnu/Linux, o Mac) pero existen algunas dificultades cuando la persona use otra plataforma, por ejemplo en Unix se usa «/» para separar directorios, mientras que en Windows se usa \ y el comando a ejecutar en el primer caso se llama `pandoc`, mientras que en el segundo se llama `pandoc.exe`.
- Podemos usar un lenguaje multiplataforma como Lua, que ya viene incluido dentro de Pandoc, de manera que nuestro *script* sea portable entre distintas plataformas tipo Unix y Windows y que pueda lidiar tanto con las diferencias entre plataformas, como con las partes repetitivas del script.

La segunda es una excelente alternativa y quizás llegaremos a ella después de hacer un recorrido más largo. Esto porque cuando se introduce a alguién a un lenguaje de programación, se le introduce realmente a toda una cultura y viajar entre culturas requiere cierta guía. Además del lenguaje en el que escribimos el código, deberemos saber qué entorno de escritura elegir, y sobre todo qué experiencia de escritura de código tener. Dentro de las muchas posibles, una de las más agradables y desconocidas es el *live coding*, que utiliza las características interactivas del computador, para convertir el acto de escritura en una especie de conversación con dicha máquina, acerca del problema que queremos abordar. Para introducirnos a esta forma de escribir, usaremos uno de los mejores entornos y lenguajes de desarrollo para *live coding*, llamado Pharo, lo cual no sólo nos presentará conceptos sencillos y poderosos de maneras interactivas, sino que nos permitirá luego revisar qué soporte para dichos conceptos y experiencias existen en otros lenguajes. Pharo se convierte entonces en una perspectiva y en una guía de navegación para la computación en sí misma, lo cual encarna la metáfora de alumbrar el camino, como hacen los faros (Pharo traduce faro en español).

Programación y *Live Coding*

Algunos afirman que la pregunta que orienta a la informática como disciplina «¿es qué podemos automatizar?» y dicha pregunta se presenta en varios campos donde los automatismos son posibles (incluida la escritura y publicación). Otros afirman que programar es esencialmente modelar y otros, que se trata de construir lenguajes de dominio específico que nos permitan expresar nuestro conocimiento sobre algún aspecto en particular. Acá veremos una aproximación que las conjuga todas y los pone en díaglo, pues ya sea que queramos automatizar, modelar o hablar de un dominio particular, tendremos que expresarlo en un lenguaje formal, que permita seguir los pasos de dicha automatización, comunicar el modelo o mirar los verbos y actores claves de ese dominio de conocimiento.

Usualmente lo que vemos en películas y currículos es personas escribiendo texto desde oscuras ventanas consolas o terminales de comandos y viendo otras cantidades de críptico texto de salida en ellas . Esta forma de trabajo es la que hemos usado a lo largo de esta obra, hasta el momento: abrimos una terminal, introducimos un comando en ella, y esperamos a que ocurra la salida. Conocemos los argumentos de dicho comando, pero no podemos explorar mucho más acerca de éste o sus argumentos. Dichos comandos son expresados en un lenguaje formal y realizan acciones: convertir a PDF una secuencia de archivos, sincronizar repositorios, etc. Algunas partes de los comandos se pueden cambiar (por ejemplo la secuencia de archivos que constituyen un PDF) y otras deben ser colocadas en determinado orden para ser entendidas por el computador (por eso se trata de un lenguaje formal, mucho más preciso, pero inflexible que los lenguajes humanos).

Ahora veremos una perspectivas a la programación y escritura de código que es diametralmente opuesta llamativa y poderosa, llamada *live coding*, que se ha traducido como programación en caliente, en vivo, o en directo, en contrate con otras formas más indirectas y «frías» de programar, escribiendo un código o comandos, usualmente convirtiendo esto a código de máquina (lo cual se conoce como compilar) y luego revisando el resultado, o imaginando lo que el programa de computo hace en nuestra cabeza y luego verificando si es así, bien sea ejecuándolo y/o colocando mensajes de impresión para ver el estado de ciertos componentes involucrados (Bret Victor dice, por esta práctica habitual de programación que se podría definir como la manipulación ciega de símbolos). El *live coding*, en contraste nos permite la exploración interactiva de los elementos que hacen parte de la experiencia de computo de maneras más

directas y ver los resultados de escribir código en la medida en que éste se va escribiendo, así como inspeccionar los componentes que hacen parte de la misam y ha sido empleada en artes performativas, como la música, pero también en temas como visualización y el activismo de datos (y ahora también para la publicación).

La música de hecho nos provee una metáfora adecuada para hablar del *live coding* pues al igual que ver una partitura o fotograías de un concierto no son sustitutos de asistir a éste, el *live coding* tiene que ser experimentado para entenderlo mejor, por un lado, y por otro, uno puede disfrutar de una experiencia de *live coding* sin saber programar, así como uno puede disfrutar de un concierto sin ser músico.

Las siguientes secciones introducirán entornos dónde aprender y practicar de live coding

Pharo y Grafoscopio

Acá deberían adaptarse las intrucciones de https://docutopia.tupale.co/pharo# y https://docutopia.tupale.co/grafoscopio#.

Principios básicos: Objetos y mensajes

Acá se debería explicar la sintaxis básica de los tres tipos de objetos que tiene, e introducir próntamente la cartilla de «Apprentice Notebook» de los documentos interactivos de Grafoscopio. La idea es que el aprendiz empiece acá, pero pase rápidamente a los textos interactivos y haga los ejercicios propuestos en la libreta para aprender la sintaxis básica del lenguaje, a fin de resolver un problema práctico.

El texto/libro como datos

Volvamos al ejemplo del subcapítulo sobre exportación en PDF:

```
cd es/
```

```
pandoc metadata.yaml \
    capitulos/presentacion.md  \
    capitulos/convenciones.md \
    capitulos/fossil.md  \
    capitulos/infraestructuras-bolsillo.md \
    capitulos/instaladores.md \
    -o documentaton.pdf
```

Veremos que el comando con el que se genera el libro tiene varias partes que se repiten una y otra vez. Por ejemplo, repetidamente estamos indicando que los capítulos se encuentran en la carpeta «`capitulos`». Podríamos resolver esto cambiando el lugar desde donde ejecutamos el *script*, de modo que en lugar de hacerlo desde la carpeta del lenguaje, lo hagamos desde la carpeta de los capítulos, con lo cual el comando de exportación sería:

```
cd es/capitulos/
```

```
pandoc ../metadata.yaml \
    presentacion.md  \
    convenciones.md \
    fossil.md  \
    infraestructuras-bolsillo.md \
    instaladores.md \
    -o ../documentaton.pdf
```

Ahora estamos indicando que, efectivamente, el comando de exportación a PDF se ejecuta desde una carpeta más interna, pero que sus parámetros de conversión y su resultado se deben guardar en una

carpeta más externa (por eso el `..\` que acompaña a `metadata.yaml` y `documentaton.pdf`).

Aún así, el orden de los archivos que conforman los capítulos y subcapítulos no es explícito, es decir, otra persona que ha descargado los archivos a su computador no sábe en qué secuencia deben ir para producir el libro, lo cual podría indicarse si cambiamos el nombre de los archivos, con lo cual la segunda parte del comando anteior; quedaría ahora

```
pandoc ../metadata.yaml \
    01-presentacion.md  \
    02-convenciones.md \
    03-fossil.md  \
    04-infraestructuras-bolsillo.md \
    05-instaladores.md \
    -o ../documentaton.pdf
```

Pero, ¿qué pasa si cambiamos el órden de los capítulos o si agregamos subcapítulos intermedios? ¿Continuaremos agregando números intermedios (ejp: `041-algo.md`, `041-algo-mas.md`) y cambiando los nombres de los archivos, tanto en el disco duro local, como en el repositorio remoto? Además, siguen habiendo repeticiones (todos los archivos terminan en `.md`) y conocimiento sobre la estructura del proyecto que no está explicitado en ningún lado, salvo en el histórico de la máquina donde ejecutamos los comandos, por ejemplo que en el comando jústamente anterior debemos ubicarnos en una carpeta en particular, que corresponde al lugar donde están los capítulos para determinado idioma. Esta «solución» prontamente se muestra inflexible y poco escalable.

Como queremos que la publicación sea automatizada, tenemos que internalizar el conocimiento que está en el entorno donde ejecutamos la conversión de formatos (en la consola de comandos del computador donde los escribimos) a un lugar que pueda viajar entre distintas máquinas sin sufrir muchos cambios, por ejemplo en un *script* de Pharo, guardando en una libreta interactiva de Grafoscopio y puesta dentro del repositorio del libro.

Lo mejor sería representar las diferentes partes del comando de Pandoc como código. Esto incluye el archivo de metatados, las carpetas que rerpresetan el lenguaje y los capítulos, así como tabla de contenido de nuestro libro, es decir la secuencia de capítulos y subcapítulos que lo conforman, el archivo de exportación y cualquier opción para el mismo. ¿Qué objeto de Pharo, de los vistos en la secciones precendentes nos puede ayudar para ello?

Cierre y continuaciones

Al final de esta parte deberías estar en condiciones de:

- ☐ Comentar algunas de las razones por las cuales es importante automatizar los proceso de publicación.
- ☐ Decir qué es el *live coding* y sus diferencias frente a otras formas de programación y escritura de código.
- ☐ Instalar Pharo y Grafoscopio.
- ☐ Conocer los rudimentos del lenguaje/entorno de programación Pharo y ejecutar *scripts* dentro del mismo.
- ☐ Organizar los *scripts* dentro de libretas interactivas de Grafoscopio.
- ☐ Usar scripts para automatizar la publicación de obras en PDF y ePub.