UNIVERSIDAD
COMPLUTENSE
MADRID

Metodología de internacionalización de material docente basada en el uso de Markdown y Pandoc

PROYECTO DE INNOVACIÓN DOCENTE

Convocatoria INNOVA-DOCENCIA 2017/2018

Proyecto Nº 38

Investigador responsable: Juan Carlos Sáez Alcaide

Facultad de Informática

Departamento de Arquitectura de Computadores y Automática

1. Objetivos propuestos en la presentación del proyecto

Este proyecto se enmarca en el contexto del Plan de Internacionalización de la Docencia de la UCM. Este plan ofrece grandes oportunidades para la universidad, pero también plantea retos importantes para estudiantes y profesores. En particular, el proyecto de innovación propone una metodología, basada en el uso de herramientas específicas, para la creación y el mantenimiento efectivo del material docente de una asignatura impartida simultáneamente en dos idiomas (diferentes grupos de estudiantes).

En la actualidad, la UCM oferta múltiples titulaciones de grado con asignaturas que se imparten simultáneamente en grupos de español y grupos en otro idioma (p.ej., inglés). De forma más general, nos referiremos a la lengua oficial mayoritaria de una universidad– el español en el caso de la UCM– como L1, y a la lengua extranjera como L2. En muchas universidades europeas encontramos escenarios bilingües similares, siendo L1, por ejemplo, el francés, el alemán o el italiano, y siendo L2 mayoritariamente el inglés. Para hacer frente a este bilingüismo más general, pretendemos que la metodología propuesta en este proyecto sea aplicable a cualquier L1 y L2.

Con frecuencia, las asignaturas impartidas en L1 y L2 tienen un alto nivel de coordinación, empleando un mismo método de evaluación para todos los grupos (examen final/prácticas comunes para todos los estudiantes). Para facilitar la coordinación en situaciones donde hay un gran número de grupos, es aconsejable mantener una cantidad significativa de material docente compartido entre profesores, como por ejemplo hojas de problemas, guiones de prácticas o presentaciones de clase. Para algunas asignaturas impartidas en la Facultad de Informática, donde imparten docencia la mayor parte de participantes del proyecto de innovación, este material compartido se ha mantenido íntegramente en L1 (español), incluso tras la implantación de los grados bilingües. La figura 1 ilustra el modelo de compartición de documentos llevado a cabo en estas asignaturas. Esencialmente, los profesores comparten documentos en distintos formatos (p.ej., presentaciones PowerPoint, documentos de Word, etc.) usando una carpeta compartida en la nube. Gracias a los servicios de sincronización de ficheros en la nube, como Google Drive o Dropbox, cualquier modificación realizada por uno de los profesores en cualquiera de los documentos se hace visible automáticamente al resto.

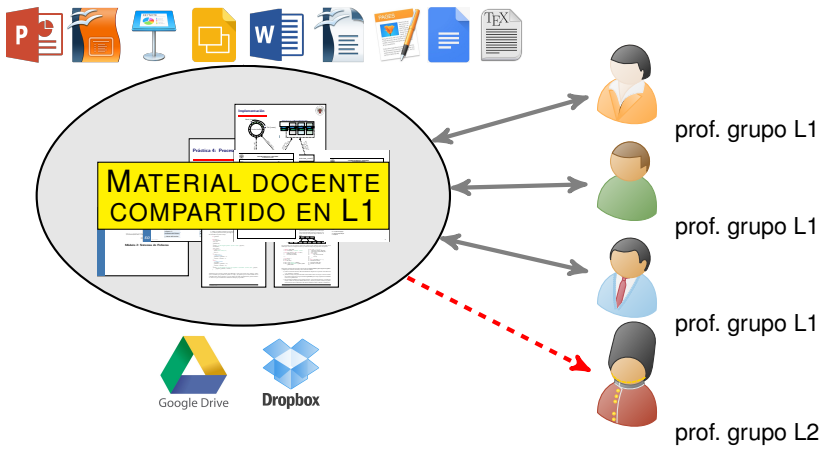


Figura 1: Compartición de material docente entre grupos en L1

Si bien esta aproximación facilita la coordinación entre los grupos en L1, el modelo de compartición de material dificulta enormemente la participación de los profesores de los grupos en L2 y en general supone una gran carga de trabajo para éstos. Esencialmente, el primer curso académico que la asignatura se imparte en L2 estos profesores han de emplear una gran

cantidad de tiempo en traducir una parte sustancial del material compartido. Además deben mantener actualizado el material, monitorizando las modificaciones –posiblemente frecuentes– que el material común en L1 sufre a lo largo del tiempo. En la práctica, esto puede ocasionar problemas de coordinación significativos derivados de la imposibilidad de incorporar a tiempo los cambios realizados en el material en L1, en su versión en L2. La gran diversidad de herramientas ofimáticas y tecnologías usadas por distintos profesores para elaborar documentación compartida dificulta aún más el mantenimiento del material.

El principal reto de este proyecto es subsanar estos problemas, y con ello permitir que los docentes de los grupos en L2 puedan dedicar más tiempo a la preparación de las clases y a la aplicación de diversas metodologías para la mejora de la calidad docente y del rendimiento académico. Para afrontar este reto proponemos una estrategia específica para la creación y mantenimiento de material docente en *dual*. La idea general es mantener en un mismo fichero de texto el contenido en L1 y L2 (p.ej., español e inglés) del documento docente que se desee construir, proporcionando justo detrás de cada párrafo y título en L1 en el documento su traducción a L2, empleando delimitadores especiales. Para crear estos documentos duales planteamos el uso de Markdown [1], un lenguaje de marcado ligero, que dada su sencillez y versatilidad está teniendo una rápida adopción por un amplio espectro de profesionales: desde escritores de novelas o periodistas, hasta investigadores o docentes [2] (p.ej., para creación de cuestionarios en Moodle). A partir de los documentos duales creados con Markdown, es posible generar automáticamente el documento final para cada idioma en el formato deseado que se pondrá a disposición de los estudiantes. Para esta tarea automática, proponemos el uso de la herramienta Pandoc [3], que permite realizar la conversión de documentos Markdown a una gran cantidad de formatos, como PDF, docx (Word), EPUB (libro electrónico) o HTML.

A la hora de diseñar nuestra metodología de creación de material docente en dual se consideraron los siguientes objetivos:

1. Garantizar que la metodología fuera aplicable a cualquier titulación de la UCM, permitiendo la creación de múltiples clases de documentos docentes (p.ej., hojas de ejercicios, guiones de prácticas, cuestionarios de Moodle, etc.) usando el lenguaje Markdown.
2. Hacer posible que la aproximación pudiera ser puesta en práctica por cualquier docente, incluso con conocimientos muy básicos de informática.
3. Asegurar que la creación de documentos duales pudiera realizarse empleando herramientas gratuitas sobre múltiples sistemas operativos.

2. Objetivos alcanzados

En este proyecto de innovación docente se han conseguido alcanzar todos los objetivos enumerados en la sección anterior. Esto ha sido posible gracias a la creación del *framework Dual Markdown*, producto desarrollado íntegramente durante el proyecto. A lo largo de la memoria, especialmente en la sección 6, proporcionaremos más detalles sobre este *framework*. Es posible obtener información adicional en la página web oficial del proyecto [4].

El *framework Dual Markdown* permite la creación de documentos duales (con contenido en L1 y L2) usando extensiones especiales del lenguaje Markdown, que se presentan en la sección 6.3. La figura 2 describe el nuevo modelo de compartición de documentos basado en el *framework Dual Markdown*. En general los profesores ya no comparten material docente en formatos convencionales como Microsoft Word o PowerPoint, sino que lo hacen usando ficheros Dual Markdown (texto plano). Para la edición de estos documentos puede emplearse cualquier editor de textos. No obstante, el editor más aconsejable para esta tarea es Sublime Text [5], editor multiplataforma gratuito para el que actualmente hay disponible un *plugin* [6] –distribuido como

parte de nuestro framework– que permite trabajar más cómodamente con documentos Dual Markdown. A partir de cualquiera de estos documentos duales compartidos –que pueden ser, por ejemplo, hojas de problemas, presentaciones de clase, etc.– cualquier docente puede generar de forma sencilla un documento final en el idioma elegido (L1 o L2) y en un formato adecuado para poner a disposición de los estudiantes. Gracias al potencial de la herramienta Pandoc, usada por nuestro framework, es posible convertir automáticamente los documentos en Dual Markdown a infinidad de formatos. Por ejemplo, un guión de prácticas construido en Dual Markdown puede exportarse a formatos como PDF (optimizado para la impresión), EPUB (optimizado para la visualización en dispositivos móviles), Word, HTML y muchos otros.

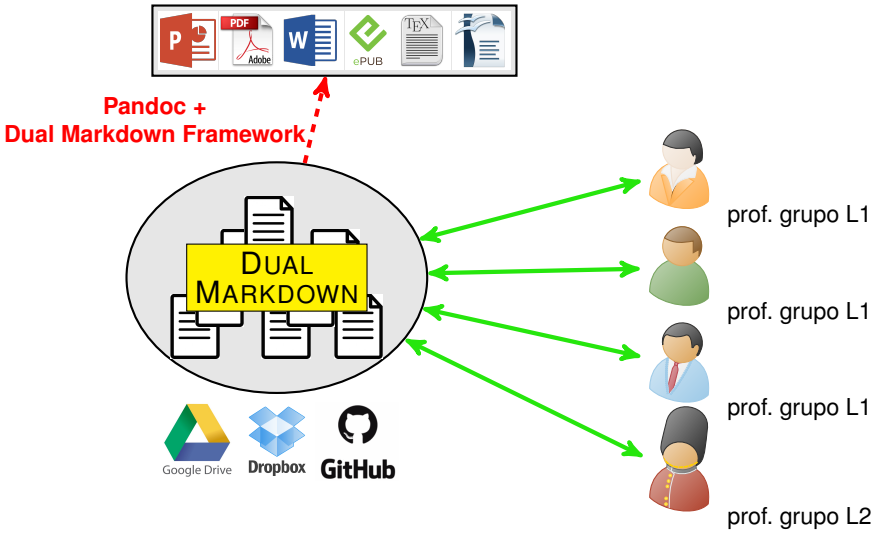


Figura 2: Compartición de material docente entre grupos en L1 y L2 usando Dual Markdown

Para ilustrar la versatilidad del *framework Dual Markdown* –primer objetivo del proyecto (ver sección 1)– hemos creado una serie de documentos de ejemplo, disponibles en Github [7]. Los ejemplos muestran cómo crear una amplia variedad de documentos docentes en formato dual e ilustran el potencial expresivo de las extensiones de Markdown creadas. En la página web del proyecto se proporcionan instrucciones [8] para trabajar con estos ejemplos.

Garantizar que los docentes con conocimientos muy básicos de informática puedan iniciarse fácilmente a la creación de documentos duales (segundo objetivo del proyecto) ha constituido un reto significativo. Si bien el aprendizaje del lenguaje Markdown y de las extensiones para creación de documentos duales no entraña dificultad dada su gran simplicidad, durante el transcurso del proyecto detectamos dos barreras significativas para los usuarios menos experimentados:

1. Generar los documentos finales a partir de documentos duales en Markdown requería el uso de comandos de Pandoc de gran longitud, y el uso del terminal del sistema operativo para invocar estos comandos.

2. El proceso de instalación manual de las múltiples herramientas que conforman nuestro *framework* (ver sección 6.1) podría resultar tedioso. Nótese que, por el objetivo 3, mencionado en la sección anterior, al elegir las herramientas a utilizar –algunas desarrolladas por terceros– se optó inicialmente por el uso exclusivo de herramientas gratuitas y multiplataforma, sin considerar de forma prioritaria la simplicidad de su instalación.

Para resolver estos problemas creamos la herramienta Panbuild [9], un *plugin* para usar Panbuild desde Sublime Text [6], y una serie de instaladores [10] que automatizan completamente la instalación de las distintas herramientas del *framework* en distintos sistemas operativos.

3. Metodología empleada en el proyecto

El desarrollo del proyecto constó de las siguientes fases:

1. Formalización de las extensiones del lenguaje Markdown para la construcción de documentos duales, y creación de extensiones adicionales para incrementar la expresividad de Markdown para la elaboración de material docente.
2. Ampliación de la funcionalidad de la herramienta Pandoc mediante los mecanismos de extensión estándar (*Pandoc Filters* [11]) para el procesamiento adecuado de las sentencias de Markdown definidas en la fase anterior.
3. Selección de las herramientas gratuitas más adecuadas para la creación de material docente con Markdown en distintos sistemas operativos.
4. Puesta en práctica de la metodología propuesta para creación de material docente en dual en asignaturas impartidas en inglés y español durante el Curso Académico 2017-2018.
5. Puesta en común de las distintas experiencias de los miembros participantes del proyecto y aplicación de las modificaciones necesarias en las herramientas y metodología desarrolladas considerando la retroalimentación obtenida. Cabe destacar que las reuniones llevadas a cabo en esta fase permitieron detectar las dos barreras críticas mencionadas en la sección 2. Esto obligó a realizar modificaciones sustanciales en la planificación del proyecto (fases que figuran a continuación).
6. Diseño e implementación de la herramienta Panbuild y creación de un *plugin* para usar esa herramienta desde el editor Sublime Text. Panbuild es actualmente un componente clave de nuestro entorno, haciéndolo más amigable y más accesible para usuarios con conocimientos muy básicos de informática.
7. Creación de instaladores para simplificar el despliegue del entorno de trabajo en distintos sistemas por parte del profesorado de la UCM.
8. Elaboración de documentación y videotutoriales para permitir la difusión y aplicación de nuestra metodología en la universidad, así como creación de la página web oficial para el proyecto [4].

Para llevar a cabo dichas fases se realizaron las siguientes tareas (ver planificación en figura 3):
T1. Diseño de las extensiones de Markdown para la construcción de documentación docente en Dual.
T2. Estudio de distintas interfaces de programación (APIs) para construir filtros de Pandoc, e implementación de filtros necesarios para procesar las extensiones de Markdown propuestas en T1.
T3. Análisis de las herramientas disponibles en el mercado para la edición de documentos Markdown en distintos sistemas operativos.
T4. Creación de manual de uso de las extensiones de markdown diseñadas en T1.
T5. Aplicación de la metodología propuesta y uso del software desarrollado en T2 para elaborar material en dual en asignaturas impartidas durante el curso académico actual.
T6. Testeo de los filtros de Pandoc desarrollados en T3, sobre Pandoc 2.0 (nueva versión).
T7. Diseño e Implementación de herramienta Panbuild y de plugin para Sublime Text.
T8. Puesta en común de las distintas experiencias docentes y aplicación de modificaciones necesarias en la metodología considerando la retroalimentación obtenida.
T9. Elaboración de la documentación del proyecto.
T10. Creación de instaladores para Windows, Linux y Mac OS X.
T11. Creación de videotutoriales de uso del Framework Dual Markdown.
T12. Creación de la página web oficial del proyecto.

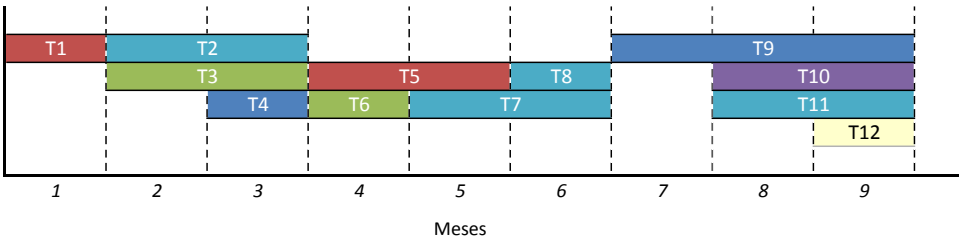


Figura 3: Planificación de las tareas del proyecto

4. Recursos humanos

En esta sección se detallan las contribuciones de cada uno de los miembros participantes del PIMCD. Cabe destacar, en cualquier caso, que todos los participantes han contribuido a la realización de las tareas T1, T5 y T8 descritas en la sección anterior –críticas para el éxito del proyecto–, por lo que esas tareas no se citan explícitamente a continuación.

Juan Carlos Sáez ha sido el investigador responsable del proyecto, y como tal, ha supervisado la realización de todas las tareas descritas en la sección anterior. En particular, ha contribuido especialmente a las tareas de diseño y desarrollo de software (T2,T7,T8 y T12), y de elaboración de documentación (T4 y T9).

Marcos Sánchez-Élez ha contribuido especialmente a la realización de las tareas T3 y T9, y ha elaborado este documento junto con el investigador responsable.

José Luis Risco ha participado activamente en la realización de las tareas T2 y T8, realizando un testeo exhaustivo de los componentes del framework Dual Markdown en distintos sistemas.

Fernando Castro ha desarrollado el instalador del framework Dual Markdown para sistemas Windows (T10) y ha contribuido a la elaboración de documentación (T9) y videotutoriales (T12).

Manuel Prieto realizó el estudio de compatibilidad para Pandoc 2.0 (T6) –nueva versión de Pandoc publicada durante el transcurso del proyecto– de las extensiones de Markdown desarrolladas.

Regino Sáez ha contribuido especialmente al testeo del instalador para Windows (T10) y a la revisión exhaustiva de la documentación elaborada (T4 y T9).

Daniel Chaver ha participado en la elaboración de documentación (T4) y en el desarrollo del instalador para Windows (T10). También ha elaborado tutoriales disponibles en la web oficial (T9 y T11).

Katzalin Olcoz ha participado en la realización de las tareas T3 y T7, realizando un testeo exhaustivo de versiones alpha de la herramienta Panbuild y de su plugin para Sublime Text.

Juan Antonio Clemente ha desarrollado el instalador del framework Dual Markdown para Mac OS X (T10).

Francisco Igual ha contribuido a la elaboración de ejemplos disponibles en la web oficial (T9) y ha testeado a fondo la funcionalidad del framework Dual Markdown en Linux.

Adrián García ha llevado a cabo el diseño y desarrollo de la web oficial del proyecto (T11), y también ha participado en la realización de las tareas T6 y T4.

David Sánchez ha realizado un análisis exhaustivo de los editores de Markdown multiplataforma (T3) y ha testeado la funcionalidad de los distintos componentes del framework en Windows 10.

5. Desarrollo de las actividades

El proyecto constó de tres etapas bien diferenciadas:

- **Etapa 1:** Durante esta etapa se procedió a la identificación de las extensiones de Markdown necesarias para la creación de documentos duales, al diseño e implementación de las mismas, a la elección de las herramientas multiplataforma para la edición y procesado de los documentos, y a la puesta en práctica de la metodología basada en Dual Markdown para la creación de material docente multiidioma.
- **Etapa 2:** Esta etapa surgió como resultado de las conclusiones obtenidas por los distintos miembros del proyecto al usar el software desarrollado en la Etapa 1 (primer prototipo del *framework*). Esencialmente, la evaluación de esta primera versión del software reveló la existencia de barreras adicionales, que nos obligaron a alterar la planificación inicial del proyecto (la que consta en el documento de solicitud) para, entre otras cosas, asignar tiempo y recursos para el desarrollo de la herramienta Panbuild y de una extensión (*plugin*) de Sublime Text para usar esta herramienta desde el editor.
- **Etapa 3:** Una vez concluido el desarrollo de las herramientas adicionales, fue posible concluir con las tareas finales del proyecto –ya planificadas inicialmente–, entre las que se incluyen la elaboración de la mayor parte de la documentación del proyecto, la creación de instaladores del framework Dual Markdown, y el desarrollo de la web oficial del proyecto.

A continuación se proporciona información más detallada sobre el desarrollo de cada etapa.

5.1. Etapa 1

Durante el primer mes del proyecto, y tras la reunión inicial a la que asistieron todos los participantes del PIMCD, se comenzó con el proceso de identificación y diseño de las extensiones del lenguaje Markdown necesarias para la creación de documentos docentes duales. En la reunión citada se decidió optar por la creación de dos clases de extensiones de Markdown.

La primera clase de extensiones, que denominamos *Dual Markdown*, abarca nuevas construcciones del lenguaje para la creación de contenido dual. Como se describe en la seccion 6.3.2, Dual Markdown soporta la creación de títulos duales, elementos de bloque y de línea duales, inclusión condicional de figuras en base al idioma elegido, etc.

La segunda clase de extensiones del lenguaje Markdown, a la que nos referiremos como *Teaching Markdown*, pretende enriquecer la expresividad de este lenguaje para la creación específica de documentos docentes. Con estas extensiones se pretende incorporar construcciones utilizadas comúnmente en documentos de carácter docente pero que el lenguaje Markdown reconocido por Pandoc no deja expresar, como es el caso de la inclusión de texto y figuras en múltiples columnas, la inclusión de saltos de página forzados, añadir texto coloreado en los documentos o controlar la alineación de figuras o párrafos. Las extensiones creadas (véase la sección 6.3.3) no solo dan una solución a este problema, sino que también ofrecen atajos, como la numeración automática de ejercicios y cuestiones con una marca especial, para simplificar la elaboración de tipos específicos de documentos docentes, como hojas de problemas.

Tras la puesta en común de las extensiones propuestas por los distintos participantes, se comenzó en paralelo a realizar dos tareas: la implementación de las extensiones de Markdown en Pandoc (T2), y el análisis de editores de Markdown existentes con soporte de Pandoc para la edición de documentos en múltiples plataformas (T3).

Antes de la implementación en sí de las extensiones de Markdown, se hizo un estudio de las interfaces de programación (APIs) disponibles para crear filtros de Pandoc [11] –mecanismo

mediante el cual creamos dichas extensiones–. Tras el estudio, se optó por utilizar la biblioteca Panflute [12] que permite la creación de filtros de Pandoc multiplataforma en Python.

La mayor parte de los miembros del proyecto que no participaron en la implementación de los filtros de Pandoc (T1-T2), ni en la elaboración del manual de uso de las extensiones (T4), realizaron un análisis de la adecuación de distintos editores de Markdown con soporte de Pandoc para la aplicación de la metodología propuesta en el proyecto. De las distintas soluciones multiplataforma y gratuitas evaluadas, los editores Typora [13] y Sublime Text [5] fueron las herramientas que resultaron más adecuadas para la creación de documentos duales. No obstante, dado que solo Sublime Text tiene la capacidad de creación de extensiones (*plugins*), apostamos finalmente por este editor para la aplicación de nuestra metodología.

Una vez acabadas las tareas T1-T4, se procedió a elaborar material docente para distintas asignaturas impartidas por los participantes del PIMCD usando el primer prototipo de nuestro *framework*, formado por las herramientas Pandoc, Sublime Text y por los filtros de Pandoc desarrollados. Paralelamente, se realizó un estudio de la compatibilidad de dichos filtros con la nueva versión de Pandoc (v2.0), que, de forma inesperada, se liberó durante el transcurso del proyecto, y que introducía numerosos cambios con respecto a la versión utilizada al desarrollar los filtros (v1.19). Afortunadamente, dada la rápida actualización de la biblioteca Panflute a Pandoc 2.0, verificamos que el código de nuestros filtros seguía funcionando correctamente.

5.2. Etapa 2

La experiencia de creación de material docente en dual usando el primer prototipo del *framework* Dual Markdown fue satisfactoria. Lamentablemente se detectaron algunos problemas críticos que, a nuestro juicio, hacían menos accesible nuestro entorno a usuarios menos experimentados. Los principales problemas detectados fueron los siguientes:

1. La instalación manual de las distintas herramientas necesarias para poder crear y procesar documentos duales podía resultar tediosa en algunos sistemas, como Windows. Nótese que además de las herramientas citadas anteriormente, para tener un entorno completamente funcional se precisaba instalar una distribución de LaTeX –para generación de ficheros PDF con Pandoc–, un intérprete de Python, Panflute [12], la herramienta `pip`, y una serie de filtros de Pandoc desarrollados por terceros, como `pandoc-crossref` [14].
2. Para construir los documentos finales (p.ej. un PDF en inglés) a partir de ficheros Dual Markdown el usuario debía usar comandos de Pandoc de bastante longitud (muchas opciones). Para ello, era preciso ejecutar los comandos en una ventana de terminal del sistema operativo o bien usando un *script* (fichero .bat en Windows o .sh en UNIX/Linux).

Para simplificar la instalación de las distintas herramientas, se tomaron dos medidas importantes. En primer lugar se optó por desarrollar (en la Etapa 3) instaladores que automatizan completamente la configuración y el despliegue de cada uno de los componentes que conforman el framework Dual Markdown, descargando automáticamente de internet aquellos componentes necesarios. En segundo lugar, para Windows y Mac OS X –los sistemas posiblemente más ampliamente usados por el profesorado de la UCM–, optamos por eliminar dependencias asociadas con el lenguaje de programación de Python, creando ejecutables a partir de las herramientas desarrolladas en este lenguaje. Esto hace posible que no sea necesario instalar –ni manual, ni automáticamente– el intérprete de Python ni la herramienta pip.

Eliminar el segundo problema arriba mencionado constituyó un importante desafío del proyecto y obligó a desarrollar software adicional, que, lamentablemente, no estaba previsto en la planificación inicial del proyecto. Dicho software debía evitar que el usuario tuviera que teclear largos comandos de Pandoc en un terminal, y preferiblemente, permitir que el usuario no tuviera que conocer a fondo la herramienta Pandoc para la construcción de documentos duales. Cabe

destacar que, a pesar de la completitud de su manual de usuario [15], Pandoc puede tener una curva de aprendizaje pronunciada.

Para satisfacer estos requisitos, desarrollamos la herramienta Panbuild [9] y un *plugin* de Sublime Text [6] para usar esa herramienta de forma transparente desde el editor[1]. Panbuild evita que el usuario tenga que introducir comandos de Pandoc en un terminal para generar cualquier tipo de documento de salida a partir de los documentos duales (con independencia del formato e idioma objetivo). Al usar esta herramienta, las opciones de pandoc para generar cada fichero de salida, y elegir el idioma y formato del documento final, se especifican de forma compacta en un fichero aparte, llamado *build.yaml*, que se codifica en el lenguaje YAML, ya usado por la herramienta Pandoc [17]. El *plugin* de Pandoc para Sublime Text permite construir el fichero *build.yaml* automáticamente (sin modificar el código YAML), y además ofrece la posibilidad de generar los distintos ficheros de salida desde la interfaz del editor, sin necesidad de usar un intérprete de comandos.

5.3. Etapa 3

Una vez concluido el desarrollo y testeo de la herramienta Panbuild, y de su *plugin* correspondiente, se procedió a la realización de las tareas finales que ya figuraban en la planificación inicial del proyecto. El principal objetivo de estas tareas era garantizar la máxima difusión del framework *Dual Markdown* y facilitar su uso a todo tipo de usuarios, mediante la elaboración de tutoriales (documentos y vídeo), manuales exhaustivos e instaladores para distintas plataformas.

La mayor parte del esfuerzo se destinó a la redacción de documentación sobre el framework –escrita en inglés para facilitar su difusión–, entre la que destaca la descripción de las extensiones de Markdown creadas [18], las instrucciones de instalación de cada una de las herramientas desarrolladas [6], [9], [10], [16] y los tutoriales ilustrando cómo usar el framework y los ejemplos disponibles [8]. Para proporcionar una descripción más intuitiva y visual de cómo usar el *plugin* de Panbuild para Sublime Text decidimos crear videotutoriales, que diseñamos empleando la herramienta Camtasia Studio. Adquirimos una licencia de esa herramienta gracias a la financiación asignada al proyecto de innovación.

Para lograr mayor difusión del framework *Dual Markdown* creamos una página web oficial para el proyecto disponible en http://dualmarkdown.org. La página web contiene la mayor parte de documentación y videotutoriales del proyecto, e incluye enlaces al resto de documentación que se encuentra en los repositorios del software desarrollado, alojados en Github. Como parte de las actividades de difusión, el investigador responsable del proyecto presentó el trabajo en el *III Seminario de Retos y Oportunidades del Aula Internacional*, celebrado en la UCM, mediante la ponencia titulada *"Creación de material docente multiidioma con Markdown para generación automática de documentos en múltiples formatos"*.

Finalmente, durante esta última etapa del proyecto se optó por desarrollar dos instaladores: uno para Windows y otro para Mac OS X. Ambos se encuentran disponibles en la página web del proyecto. Para desarrollar estos instaladores, se empleó NSIS [19] y Packages [20]. Para el testeo de los mismos, se crearon máquinas virtuales usando un ordenador portátil con Mac OS X (principal plataforma de desarrollo) usando VMware Fusion –software comercial que adquirimos con los fondos del PIMCD–. Dado que la mayor parte de los usuarios de GNU/Linux tienen conocimientos básicos sobre el uso del terminal, optamos por no crear un instalador específico para este sistema operativo. Para desplegar las distintas herramientas de nuestro framework en GNU/Linux es preciso seguir las instrucciones de instalación manual descritas en la página web del proyecto [10].

[1] Ambas herramientas son libres, y su código fuente está disponible gratuitamente en Github, al igual que el resto del software desarrollado en el proyecto [16].

6. Anexos

En esta sección final se describen los componentes del *framework* Dual Markdown, se presenta un documento Dual Markdown de ejemplo –indicando también cómo generar documentos finales en un idioma y formato específico a partir de él–, y se describen las extensiones de Markdown diseñadas e implementadas en este proyecto.

6.1. Componentes del Framework Dual Markdown

Nuestro framework consta de los siguientes componentes multiplataforma y *open source*, desarrollados durante el transcurso del proyecto:

1. **Filtros de pandoc de Dual Markdown** [16]. Estos filtros extienden la variante de Markdown reconocida por Pandoc con características de creación de documentos docentes duales. En la sección 6.3 se describen las extensiones de Markdown que implementan estos filtros.
2. **Panbuild** [9] es una herramienta de línea de comandos (inspirada en GNU Make, aunque mucho más simple) que facilita al usuario la interacción con Pandoc, permitiendo que éste especifique las opciones de Pandoc necesarias para construir cada tipo de fichero de salida (PDF, DOCX, etc.) en un fichero aparte (*build.yaml*) usando el lenguaje YAML.[2] En la documentación oficial de Panbuild [21] se describe la sintaxis del fichero *build.yaml*.
3. **Plugin de Panbuild para Sublime Text** [6]. Este plugin hace posible interactuar de forma sencilla con Panbuild desde la interfaz del editor Sublime Text, evitando que el usuario tenga que crear el fichero *build.yaml* manualmente, y permitiendo la generación automática de distintos documentos con Pandoc sin tener que usar comandos en una ventana de terminal. En la sección 6.2 se ilustra el potencial de este *plugin*

Para poder usar el framework Dual Markdown en Windows, Linux o Mac OS X se precisa la instalación de los componentes anteriormente citados, así como de las siguientes herramientas gratuitas y multiplataforma desarrolladas por terceros:

- **Pandoc** [3]. Se trata de un conversor universal de documentos, usado en nuestro proyecto para generar ficheros PDF, Word, EPUB, etc. automáticamente a partir de documentos Dual Markdown.
- **Pandoc-crossref** [14]. Filtro (extensión) de Pandoc que permite incluir referencias a figuras, tablas, ecuaciones y secciones de un documento Markdown.
- El editor **Sublime Text 3** [5] permite la edición de ficheros Dual Markdown.
- **Una distribución de LaTeX** [22]. Necesaria para poder generar ficheros PDF con Pandoc.

Todas estas herramientas se instalan de forma completamente automatizada al usar el instalador del framework Dual Markdown, disponible para Windows y Mac OS X. Para la instalación en Linux, es preciso seguir las instrucciones de instalación manual [4].

[2] El lenguaje YAML se usa en Pandoc para especificar distintos aspectos que se tienen en cuenta durante la construcción de un documento, como el fichero que contiene las referencias bibliográficas, aspectos de estilo (p.ej., tamaño de página o de fuente a utilizar) o los metadatos del fichero generado. Se optó por elegir el lenguaje YAML en Panbuild por el hecho de que los usuarios de Pandoc ya están familiarizados con él.

6.2. Ejemplo de documento Dual Markdown

Para mostrar la versatilidad de las extensiones de Markdown creadas, elaboramos una serie de documentos de ejemplo disponibles en Github [7]. Los ficheros de ejemplo pueden descargarse como un único fichero ZIP usando este enlace. Al extraer en ZIP, se crearán varias carpetas, una por cada ejemplo, entre las cuales se encuentra el directorio `hello`, ejemplo que describiremos en esta sección.

El ejemplo *hello* consta de un fichero Dual Markdown (`hello.md`) y del archivo `build.yaml`, que contiene la información necesaria para que Panbuild genere múltiples documentos de salida a partir del contenido del fichero `hello.md`. Aquí describiremos únicamente la estructura del fichero Dual Markdown y proporcionaremos instrucciones sobre cómo generar ficheros en distintos formatos a partir de él usando Panbuild desde Sublime Text o desde la línea de comandos del sistema operativo. Nótese que el fichero *build.yaml* puede generarse automáticamente y de forma sencilla usando Sublime Text (veáse el videotutorial presente en la web del proyecto [8]). Para obtener más información sobre la sintaxis del fichero *build.yaml*, es preciso consultar la documentación de Panbuild [21].

6.2.1. Descripción del ejemplo

El contenido del fichero `hello.md` se muestra a continuación:

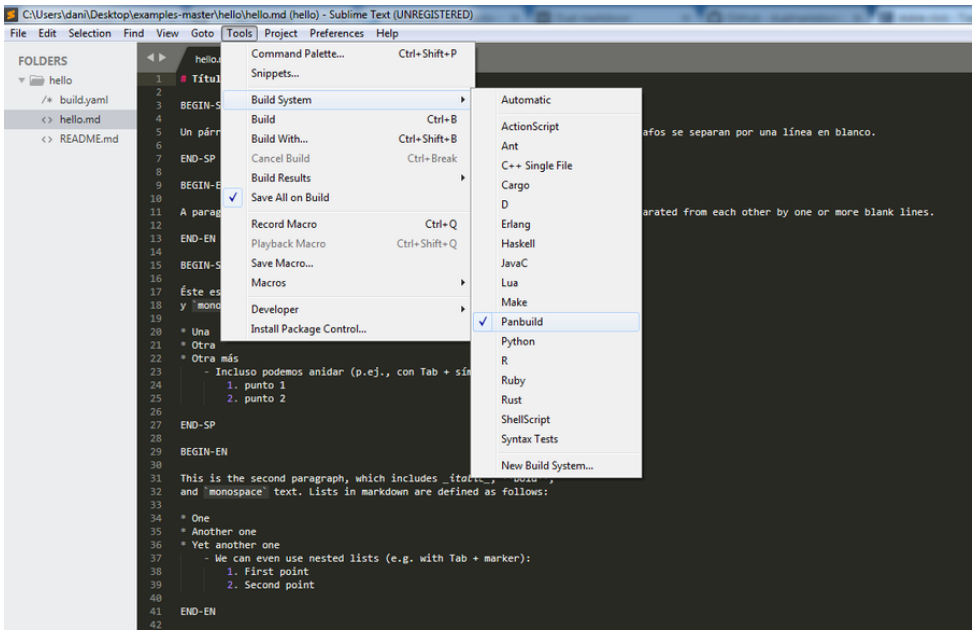
```
# Título de nivel 1 ||| Level-1 title

BEGIN-SP

Un párrafo está formado por una o varias líneas de texto consecutivas. Los párrafos se separan
por una línea en blanco.

END-SP

BEGIN-EN

A paragraph is simply one or more consecutive lines of text. Paragraphs are separated from each
other by one or more blank lines.

END-EN

BEGIN-SP

Éste es el segundo párrafo, que incluye *cursiva*, **negrita**, y `monospace`. Las listas en
Markdown se definen como sigue:

* Una
* Otra
* Otra más
    - Incluso podemos anidar (p.ej., con Tab + símbolo):
        1. punto 1
        2. punto 2

END-SP

BEGIN-EN

This is the second paragraph, which includes _italic_, **bold**, and `monospace` text. Lists in
Markdown are defined as follows:

* One
* Another one
* Yet another one
    - We can even use nested lists (e.g. with Tab + marker):
        1. First point
        2. Second point

END-EN
```

Este documento de ejemplo ilustra el uso de la sintaxis para bloques duales y para títulos duales, que se describe en detalle en la sección 6.3, donde se documentan también el resto de extensiones. Esencialmente el documento consta de un único encabezado de nivel 1, donde se proporciona el título en español (“Título de nivel 1”) y su traducción al inglés (“Level-1 title”), ambos separados por “|||”. En el documento cada párrafo en español –delimitado por `BEGIN-SP` y `END-SP`– tiene a continuación su traducción al inglés –encerrada entre `BEGIN-EN` y `END-EN`. Nótese que los delimitadores `BEGIN-` y `END-` asociados a cada idioma tienen como sufijo la etiqueta (*tag*) establecida para dicho idioma. En la sección 6.3.2 se proporciona información sobre cómo establecer esas etiquetas.

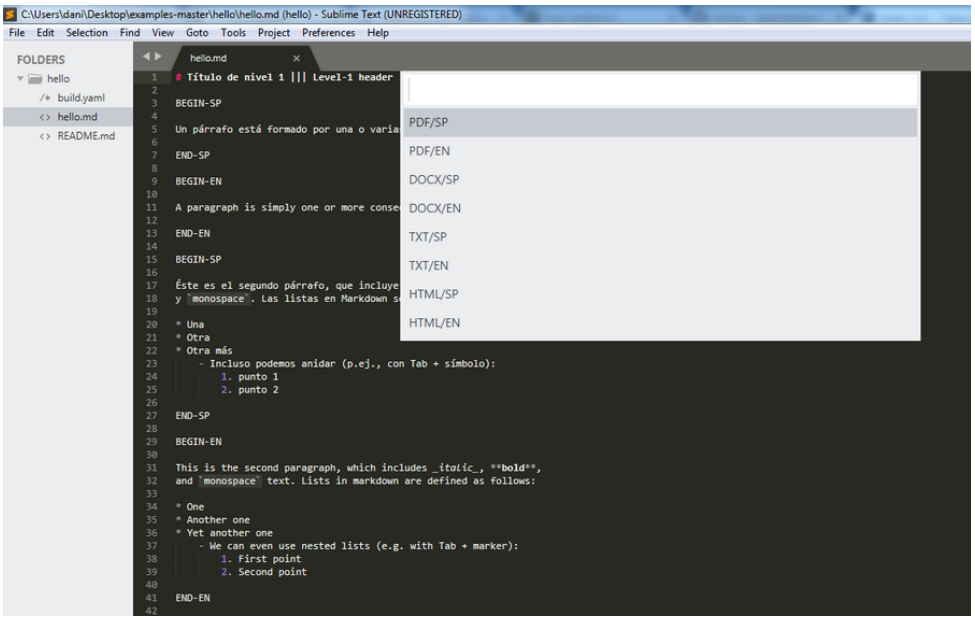
6.2.2. Usando Sublime Text para generar documentos a partir del ejemplo

Para construir documentos en distintos formatos de salida a partir del fichero Dual Markdown del ejemplo, es posible usar el editor Sublime Text con el *plugin* de Panbuild, que es parte integral de nuestro framework. En esta sección indicamos los pasos necesarios para ello.

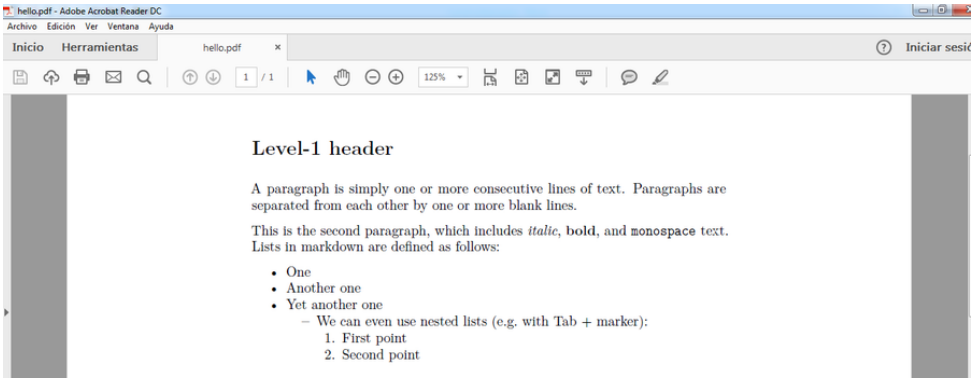
En primer lugar es preciso abrir el directorio o carpeta `hello`, que contiene los ficheros del ejemplo, con Sublime Text, para lo cual arrastraremos la carpeta a la ventana del editor. A continuación abriremos el fichero *hello.md* con el editor, por ejemplo, haciendo doble clic en la entrada correspondiente al fichero presente en la barra lateral izquierda donde se muestra el directorio `hello`. Ahora procederemos a habilitar el sistema de compilación (*build system*) de “Panbuild” haciendo clic en la siguiente entrada de menú, como se indica en la figura: *Tools– Build System– Panbuild*.



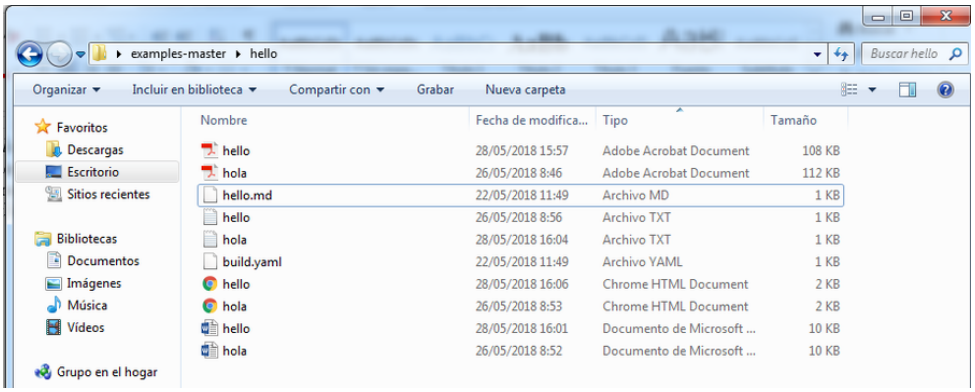
Para indicar a *Panbuild* que transforme el documento Markdown en uno de los formatos de salida definidos en el fichero *build.yaml*, basta con teclear `CTRL+B` (en Windows o Linux) o `CMD+B` (en Mac OS X). Al hacer eso, se mostrará una lista con los formatos de salida disponibles –PDF, DOCX, TXT y HTML, en este caso– y con los idiomas disponibles –español (SP) e inglés (EN)–, como se muestra en la siguiente figura:



Al hacer clic en la entrada de la lista que deseemos (formato/idioma a nuestra a elección), Panbuild iniciará el proceso de conversión del documento, invocando Pandoc. Acabada la conversión, el fichero generado se abrirá automáticamente con la aplicación predeterminada para visualizar ese tipo de fichero (p.ej., Adobe Reader para archivos PDF):



Nótese que el fichero de salida generado, que será diferente para cada formato e idioma, se ubicará en el mismo directorio donde se encuentre el fichero `build.yaml`:



6.2.3. Generación de documentos de salida usando un terminal

También es posible convertir el documento Dual Markdown a diferentes formatos de salida empleando el intérprete de comandos del sistema operativo. Para ello, es preciso abrir una ventana del intérprete de comandos o terminal del sistema, y cambiar al directorio `hello` (con `cd hello`). Una vez ubicados en ese directorio, se puede invocar el programa `panbuild` de distintas formas:

1. Tecleando únicamente `panbuild` en la línea de comandos (sin ninguna opción), se generarán todos los posibles documentos de salida, tal y como se definan en el fichero *build.yaml*, que en el caso del ejemplo son los siguientes: *hello.docx*, *hello.html*, *hello.pdf*, *hello.txt*, *hola.docx*, *hola.html*, *hola.pdf* y *hola.txt*.
2. Tecleando `panbuild <formato>/<idioma>`, el programa construirá solamente el documento en el formato e idioma especificado. Por ejemplo, el comando que se muestra a continuación creará el fichero *hello.docx* (versión en inglés del documento en formato docx de Microsoft Word):

   ```
   panbuild DOCX/EN
   ```

6.3. Extensiones de Markdown

Las extensiones creadas en este proyecto se realizaron sobre la variante de Markdown reconocida por *Pandoc*, y, se agrupan en dos clases, a las que nos referiremos como *Dual Markdown* y *Teaching Markdown* respectivamente. Las extensiones de la primera clase se han diseñado específicamente para la creación de documentos duales. Los idiomas por defecto para los documentos duales son español e inglés. En cualquier caso, es posible seleccionar otros idiomas simplemente definiendo variables específicas de Pandoc, como se indica en la sección 6.3.2. Las extensiones en la segunda categoría abarcan principalmente construcciones que se usan comúnmente al crear material docente, pero que, desafortunadamente, no son compatibles con la variante de *Markdown* reconocida por *Pandoc*. Por lo tanto, el objetivo principal de estas extensiones es incluir las características clave no soportadas en el lenguaje original, y garantizar que pueden traducirse correctamente a los formatos de salida más utilizados en el contexto académico (PDF, DOCX, EPUB, etc.).

6.3.1. Elementos genéricos de bloque y de línea en Markdown

La gran mayoría de las extensiones creadas sobre Pandoc's Markdown utilizan las construcciones genéricas para elementos de línea y de bloque ya existentes. Los elementos HTML `<span>` y `<div>` se pueden incluir en cualquier documento Markdown, por lo que se pueden usar como contenedores genéricos. En los siguientes enlaces se puede obtener más información sobre estos elementos HTML:

- https://www.w3schools.com/tags/tag_span.asp
- https://www.w3schools.com/tags/tag_div.asp

En *Markdown*, estos elementos pueden tener tres tipos de atributos:

- `class`: que se usa principalmente para especificar el tipo de elemento de línea o bloque. Se pueden indicar múltiples clases proporcionando una lista de valores separados por

espacios, como en el siguiente ejemplo:[3]

```
<div class="myclass cool_class my_paragraph">
  ... Markdown code ...
</div>
```

- `id`: para especificar identificadores únicos en el documento
- *generic attributes*: Además de `class` e `id`, podemos agregar cualquier atributo que deseemos. No obstante, solo tiene sentido incluir los denominados atributos HTML globales al generar ficheros de salida basados en HTML, como por ejemplo, EPUB o reveal.js. Cabe destacar que para la implementación de extensiones de *Markdown* podemos incluir atributos arbitrarios para ser procesados por *Pandoc* o por cualquier filtro de *Pandoc*.

Para los elementos genéricos de línea (*span*), Pandoc reconoce una sintaxis alternativa más apropiada para Markdown, que se reduce a incluir el texto *inline* entre corchetes; los atributos deben especificarse, a continuación, entre llaves.

Sintaxis:

```
[... Markdown text ...]{.class1 .class2 #id attr1=val1 attr2=val2 ...}
```

Nótese que el texto `atributo=valor` dentro de las llaves nos permite especificar *atributos genéricos*. Sin embargo, los nombres de clases e IDs, deben ir precedidos de `'.'` y `'#'` respectivamente, como se indica en el manual de Pandoc [15].

Ejemplo:

```
[My underlined text]{.underline}
```

Esto es equivalente a:

```
<span class="underline">My underlined text</span>
```

6.3.2. Dual Markdown

Las extensiones *Dual Markdown* permiten al usuario definir elementos básicos (texto, imágenes, párrafos, títulos, etc.) asociados con cualquiera de los dos idiomas configurados para el documento de *Dual Markdown*. De ahora en adelante, nos referiremos al primer idioma como *L1*, y al segundo como *L2*.

Téngase en cuenta que los dos idiomas en un documento *Dual Markdown* tienen una etiqueta (*tag*) asociada. Por defecto en *Dual Markdown*, *L1* es español, y *L2* inglés. Las etiquetas predeterminadas para estos dos idiomas son `SP` y `EN`, respectivamente. Cambiar la configuración del idioma implica redefinir las variables Pandoc `lang1` y`lang2`, que almacenan las etiquetas asociadas con L1 y L2, respectivamente. El valor de estas variables se puede establecer a través de la opción `-M` en la línea de comandos de Pandoc o incluyendo la definición de la variable en el encabezado YAML del archivo principal de Markdown del documento.[4] Nótese que las etiquetas de idioma no están preestablecidas; es totalmente decisión del usuario elegir qué etiqueta usar para un idioma en particular.

Al invocar `pandoc` para generar un documento de salida (p.ej., PDF o DOCX) a partir de un fichero *Dual Markdown*, sólo se habilitará uno de los dos idiomas. A menos que se indique

[3] Los ejemplos de esta sección están en inglés porque la documentación oficial del framework Dual Markdown se ha redactado íntegramente en inglés para facilitar su difusión.

[4] Si se usa Panbuild –opción recomendada– para generar los documentos de salida a partir de ficheros Dual Markdown, estas etiquetas han de definirse en el fichero *build.yaml*.

explícitamente, L1 –cuya etiqueta está definida en la variable `lang1`– es el lenguaje seleccionado por defecto. Para habilitar un idioma específico, la variable `lang_enabled` se debe definir asignándole la etiqueta del idioma deseado (p.ej., `lang_enabled=EN` para inglés).

La siguiente tabla resume las extensiones de Markdown para la creación de documentos duales. Estas extensiones se implementan en el filtro de Pandoc `dual_md` disponible aquí.

Extensiones	**Formatos de salida soportados**
Bloques específicos de idioma	Todos
Elementos de línea duales	Todos
Títulos duales	Todos
Celdas de tabla duales	Todos
Imágenes duales	Todos
Inclusión de diagramas Tikz como figuras	Basados en LaTeX
Diagramas Tikz Duales	Basados en LaTeX
Variables de Pandoc duales	Todos
Referencias cruzadas duales con `pandoc-crossref`	Todos

Bloques específicos de idioma

Esta extensión permite al usuario especificar los bloques de Markdown asociados con L1 o L2. Cuando se genera el documento final, el bloque en cuestión solo se mostrará si se selecciona el idioma correspondiente.

Sintaxis para bloques específicos de idioma:

```
BEGIN-<LANGUAGE_TAG>

... Markdown text block ...

END-<LANGUAGE_TAG>
```

Ejemplo:

```
BEGIN-SP

Días de la semana:

* lunes
* martes
* miércoles
...

END-SP

BEGIN-EN

Days of the week:

* Monday
* Tuesday
* Wednesday
...

END-EN
```

Elementos de línea duales (*Dual spans*)

La característica *Dual spans* permite proporcionar una versión dual de un fragmento de texto Markdown en línea, proporcionado tanto en L1, como en su traducción a L2. Como sucede con los bloques específicos del idioma, el documento final solo mostrará la versión del texto en línea que corresponde al idioma habilitado al invocar `pandoc`.

Sintaxis:

```
[<Markdown Text for L1> ||| <Markdown Text for L2>]{.dual}
```

Sintaxis alternativa:

```
[<Markdown Text for L1> ;;; <Markdown Text for L2>]{.dual}
```

Ejemplo:

```
[Módulo 2: Sistemas de Ficheros ||| Unit 2: File Systems]{.dual}
```

Títulos duales

Los elementos de línea duales se pueden usar en encabezados como en el siguiente ejemplo, que usa un encabezado de nivel 3:

```
### [Título de sección ||| Section title]{.dual}
```

El título dual constituye una versión simplificada menos verbosa de la sintaxis para *dual spans*, y específica para encabezados. Empleando la sintaxis de títulos duales, el ejemplo anterior se puede reescribir de la siguiente manera:

```
### Título de sección ||| Section title
```

La cadena `";;;"` también se puede utilizar como separador en lugar de `"|||"`:

```
### Título de sección ;;; Section title
```

Celdas de tabla duales

Las celdas de tabla en duales constituyen una versión simplificada de la sintaxis de *dual spans* específica para celdas de tabla que están formadas por un elemento de Markdown en línea. Este tipo de celdas pueden usarse, por ejemplo, en los encabezados de la tabla, como se indica a continuación:

```
| **Nombre** ;;; **Name** | **Tipo** ;;; **Type** | **Fecha** ;;; **Date** |
| :--------------------: | :-------------------: | :--------------------: |
|          DATA1         |           F           |         8-2-05         |
|          DATA2         |           F           |         8-3-06         |
|                        |                       |                        |
```

En el ejemplo, se proporciona una versión del encabezado de la tabla mediante el uso de celdas de tabla duales. Téngase en cuenta que en este contexto sólo está permitido el separador `";;;"` para evitar la ambigüedad sintáctica en las tablas tipo *pipe* de Pandoc.

Imágenes duales

Esta extensión constituye una forma simple de incluir la versión específica del idioma de una imagen (por ejemplo, versión en inglés o español) en el documento final con una sola línea de código Markdown. Para usar esta característica, se deben proporcionar dos versiones de la figura (para L1 y L2). Además, los nombres de los archivos de figuras deben incluir la etiqueta de idioma, que se omitirá cuando se use una declaración de inclusión de imagen dual.

Para ilustrar esta característica, consideremos el siguiente ejemplo de una declaración de imagen dual:

```
![](images/figure.pdf){.dual}\
```

Como puede observarse, se trata de declaración de inclusión de imagen normal (sin título) pero con la clase `dual`. Al procesar esta sentencia, el analizador *Dual Markdown* incluirá la figura con el nombre `images/figure-<LANG_TAG>.pdf`, donde `<LANG_TAG>` indica la etiqueta del idioma habilitado al invocar `pandoc`. Por ejemplo, si el idioma español –con la etiqueta `SP`– está habilitado, el texto Markdown anterior será reemplazado por el siguiente texto:

```
![](images/figure-SP.pdf)\
```

Inclusión de diagramas Tikz como figuras

En la variante de Markdown de Pandoc no se admite la inclusión de figuras creadas con el paquete Tikz de LaTeX. Gracias a esta nueva característica, las figuras con la extensión `".tex"` serán tratadas como una figura de Tikz e incluidas apropiadamente en el documento final, que debe ser PDF, LaTeX o Beamer. Si la clase `standalone` se especifica en la declaración de inclusión de imagen, el analizador de *Dual Markdown* incluirá la figura en el documento final utilizando la macro`\includestandalone{...}`. En caso contrario, la imagen se incluirá con una declaración LaTeX `\input{...}`.

También merece la pena señalar que esta extensión de Markdown se puede utilizar junto con la extensión de imágenes duales presentada anteriormente.

Diagramas Tikz duales

Otra característica interesante del framework Dual Markdown es la capacidad de crear *Diagramas Tikz duales*. Un diagrama de este tipo es una figura de Tikz que proporciona la traducción (en L1 y L2) para cada fragmento de texto que aparece en la figura. En otras palabras, los diagramas Tikz duales constituyen una forma de definir dos versiones del mismo diagrama (para L1 y para L2) en un solo archivo. Este tipo de diagramas se debe crear utilizando la macro `\dtext`, que se define automáticamente al procesar un archivo *Dual Markdown*. Esta macro acepta dos argumentos obligatorios: el primero es la versión del texto en L1, el segundo corresponde a la traducción del texto en L2.

Consideremos la siguiente sentencia, incluida en un diagrama hipotético *Dual Tikz*:

```
\dtext{Texto en español}{Text in English}
```

Al generar el documento final para L1, se mostrará `"Texto en español"` en el diagrama correspondiente. Análogamente, la cadena `"Text in English"` se mostrará en el diagrama cuando se habilite L2 al construir el documento.

Variables de Pandoc duales

Pandoc's Markdown permite al usuario definir algunas partes del documento, como el título, el subtítulo o la fecha, mediante variables pasadas como parámetro en la línea de comandos (opción -M) o mediante fragmentos de código YAML en los ficheros Markdown. Ya que el título y el subtítulo de un documento probablemente serán diferentes en L1 y L2, tiene sentido definir las variables correspondientes de forma diferente en función del idioma habilitado en el documento de destino. Para hacer esto posible, *Dual Mardown* introduce una sintaxis simple que permite definir las dos variantes del título y del subtítulo (para L1 y L2, respectivamente), usando los separadores `"|||"` o `";;;"` , como en los títulos duales.

Por ejemplo, el siguiente fragmento de código YAML podría usarse en el encabezado YAML de un documento *Dual Markdown*:

```
---
title: "Este es el título en español ||| This is the title in English"
subtitle: "Mi subtítulo ||| My subtitle"
...
```

Referencias cruzadas duales con `pandoc-crossref`

El filtro `pandoc-crossref` [14] extiende la variante de Markdown soportada por Pandoc con construcciones para hacer referencia a imágenes, ecuaciones y tablas. Este filtro también permite al usuario controlar cómo se representan las referencias y los títulos de los diversos elementos en el documento final. Personalizar este aspecto se reduce a definir distintas variables de Pandoc; para obtener más información sobre esto, se puede consultar este enlace). Por ejemplo, las variables `figureTitle` y `figPrefix` de `pandoc-crossref` nos permiten definir la etiqueta asociada con las leyendas de las imágenes y las referencias de las imágenes, respectivamente. Los valores predeterminados para estas variables son “Figure” y “fig.” respectivamente.

Dual markdown emplea `pandoc-crossref` para soportar la definición de referencias a imágenes o tablas, y, lo que es más importante, amplía la funcionalidad de este filtro de Pandoc con soporte para documentos duales. Más concretamente, el framework *Dual Markdown* permite al usuario proporcionar la versión L1 y L2 para las diversas etiquetas personalizables expuestas por `pandoc-crossref`. Para hacer esto posible, las distintas variables `pandoc-crossref` relacionadas con cada etiqueta ahora pueden tener un valor “dual” usando la misma sintaxis que la usada para variables duales de Pandoc discutidas anteriormente.

Ejemplo:

```
---
figureTitle: "Figura ||| Figure"
tableTitle: "Tabla ||| Table"
...
```

El analizador sintáctico de las extensiones *Dual Markdown* se encarga de seleccionar el valor “correcto” para las diferentes variables y las pasa al filtro `pandoc-crossref`. Una implicación importante de este comportamiento es el hecho de que el filtro `dual_md` debe especificarse antes que el filtro `pandoc-crossref` en la línea de comando de Pandoc.

7. Extensiones para docencia: *Teaching Markdown*

La tabla que se muestra a continuación resume las extensiones *Teaching Markdown*, implementadas en el filtro de pandoc `teaching_md` disponible aquí.

Extensiones	**Formatos de salida soportados**
Imágenes multiformato	Todos
Bloques de LaTeX con código Markdown	Basados en LaTeX
Tablas de LaTeX tradicionales	Basados en LaTeX
Bloques sombreados y enmarcados	Basados en HTML o LaTeX
Delimitadores de ejercicios y numeración automática	Todos
Inclusión de múltiples columnas	Docx, Basados en HTML o LaTeX
Saltos de página	Docx, Basados en HTML o LaTeX
Elementos de línea adicionales (*underline*, *alert*, *color*)	Docx, Basados en HTML o LaTeX
Modificadores para el control del tamaño de fuente	Basados en HTML o LaTeX
Alineación de figuras y de bloques	Basados en HTML o LaTeX
Bloques de Beamer avanzados	Beamer
Animaciones de Beamer	Beamer

Imágenes multiformato

Pandoc permite la creación de múltiples documentos de salida a partir de un único fichero fuente en Markdown. Si dicho fichero fuente incluye imágenes, debemos garantizar que el formato de las mismas sea adecuado para el formato de salida. Por ejemplo, si queremos generar HTML a partir de Markdown, solo se permite la inclusión de imágenes SVG, JPEG o PNG. Por el contrario, las imágenes en PDF son más adecuadas cuando se genera una salida Docx o PDF (basada en LaTeX). Lamentablemente, *Pandoc´s Markdown* carece de soporte para especificar la inclusión condicional de imágenes en las fuentes para distintos formatos de salida. Esto es necesario para poder generar correctamente distintos tipos de documentos a partir del mismo fichero fuente en Markdown.

Las sentencias de inclusión de imágenes multiformato permiten al usuario indicar qué formato de imagen usar (como *png* o *pdf*) para una figura específica, y dependiendo del tipo de documento de salida que se esté generando.

Sintaxis:

```
(path to image){alt-ext=fig-format1/.doc-formatN,...,fig-formatN/.doc-formatN}
```

Para ilustrar cómo funciona esta extensión, consideremos el siguiente ejemplo. Supongamos que tenemos una figura de Tikz (fichero *structure.tex*) –localizado en el directorio *img*-, que usaremos para generar el documento en cualquier formato a excepción de *html* y *docx*. En estos dos casos especiales, usaremos las figuras *structure.png* y *structure.pdf* respectivamente, ambas ubicadas también en el directorio *img*.

```
![](img/structure.tex){alt-ext=html/.png,docx/.pdf align=center}\
```

Bloques de LaTeX con código Markdown

Pandoc Markdown permite al usuario incluir bloques LaTeX delimitados entre `\begin{..}` y `\end{...}`. Al generar una documento de salida con un formato no basado en LaTeX (p.ej., DOCX o HTML) estos bloques simplemente son eliminados por Pandoc y no aparecerán en el fichero de salida. Por el contrario, para documentos de salida basados en LaTeX (por ejemplo, PDF, Beamer, etc.) Pandoc carga estos bloques LaTeX directamente en el documento de salida sin procesamiento adicional, para que sean gestionados directamente por el procesador de LaTeX. Esto supone una limitación importante: no es posible agregar código Markdown dentro de bloques LaTeX `\begin{..} - \end{...}`, ya que se identificará por Pandoc como código LaTeX. Para abordar este problema se han propuesto varias soluciones que se basan en el uso de macros especiales de LaTeX. Lamentablemente todas estas soluciones presentan deficiencias.

Para superar esta limitación, *Teaching Markdown* permite incluir código Markdown dentro de los bloques LaTeX utilizando `\lbegin{..}` y `\lend{...}`, en lugar de `\begin{..}` y `\end{...}`. Cuando el filtro de pandoc `teaching_md` detecta líneas con `\lbegin{..}` y `\lend{...}` transforma estas líneas en el correspondiente `\begin{..}` y `\end{...}` de LaTeX de forma automática. Al hacerlo, el código Markdown dentro de `\lbegin{..}` y `\lend{...}` ahora se procesa de forma efectiva por Pandoc y es transformado al el formato de salida correspondiente.

Como aplicación interesante de esta extensión de Markdown, el siguiente fragmento de texto permite ampliar los márgenes de página para usar en una diapositiva de Beamer individual:

```
## Slide title

\lbegin{adjustwidth}{-2em}{-2em}

... MARKDOWN CODE GOES HERE...

\lend{adjustwidth}
```

Tablas de LaTeX tradicionales

Esta extensión (habilitada de manera predeterminada) sólo afecta a la forma en la que las tablas se generan en formatos basados en Latex o basados en HTML. Simplemente se encarga de mostrar todas las tablas en modo retícula (*grid*): las filas y las columnas están separadas por líneas horizontales y verticales. Esto contrasta con el estilo predeterminado de representación de tablas utilizado por Pandoc, que no siempre es la opción preferida por los usuarios (véase este enlace).

Bloques sombreados y enmarcados

Esta extensión permite al usuario definir un bloque de texto sombreado o enmarcado en Markdown, utilizando un elemento de bloque genérico con la clase *shaded* o *framed* respectivamente.

Sintaxis:

```
<div class="shaded">
... Markdown text block goes here ...
</div>
```

Ejemplo:

```
<div class="shaded">

What is the capital city of Spain?

a. Barcelona
b. Madrid
c. La Coruña
d. Sevilla

</div>
```

Delimitadores de ejercicio y numeración automática

Esta extensión permite al usuario delimitar y enumerar un conjunto de ejercicios (marcador `exercise`), cuestiones (marcador `question`) o un elemento personalizado en el documento mediante contenedores genéricos de línea.

Cuando incluimos un nuevo marcador `exercise`, el texto del ejercicio anterior finaliza y comienza el texto del próximo ejercicio, precedido por el siguiente número entero en orden secuencial. De manera similar, el marcador `question` delimita dos cuestiones e incluye el siguiente entero secuencial. Restablecer la numeración en algún punto también es posible incluyendo `".reset"` (clase) en el delimitador de ejercicio o de cuestión. Nótese que los ejercicios y las preguntas se enumeran de forma independiente.

Sintaxis de delimitador de ejercicio:

```
### {.exercise}
```

Sintaxis de delimitador de cuestión:

```
### {.question}
```

Ejemplo:

```
# Exercises { .unnumbered }

### {.exercise}

Explain briefly *The French Revolution*.

# Questions { .unnumbered }

### {.question}

Who wrote *El Quijote*?

### {.question}

Which is the common root for Spanish, French and Italian languages?
```

Para numerar otro tipo de elementos del documento con una etiqueta personalizada, se puede usar el siguiente tipo de extensión en el documento.

```
[<Label> #]{}
```

donde `<Label>` puede ser cualquier cadena de caracteres (etiqueta). La primera vez que una etiqueta particular se utilice en el documento con esa declaración, se creará un contador y se inicializará a 1 en el código del filtro. Las apariciones posteriores del mismo elemento de línea (con la misma etiqueta) provocarán un incremento del contador. El elemento de línea se representará en el documento final de la siguiente manera: `<Label> <CurrentCounterValue>`. Además, ".reset" se puede usar dentro de las llaves del contenedor de línea para restablecer el contador personalizado cuando sea necesario.

Cabe destacar que el uso del símbolo "#" para crear e incrementar contadores personalizados se inspiró en la sintaxis del filtro `pandoc-numbering` [25], que proporciona una gama más amplia de formatos de numeración.

Inclusión de múltiples columnas

Esta extensión permite dividir una región específica del documento en varias columnas. Para ello, se debe usar un elemento genérico de bloque con las clases `columns` y `column`, como se explica a continuación.

Toda la región que se dividirá en múltiples columnas debe estar encerrada en un bloque `<div>` con la clase `columns`. A su vez, dentro de la región de varias columnas, el texto que pertenece a cada columna debe incluirse en un bloque `<div>` con la clase `column`, junto con el atributo `width`, que indica el ancho de la columna. A cada columna se le puede asignar un ancho diferente.

Existen otros atributos de relevancia que conviene mencionar. Por ejemplo, el atributo `colsep` utilizado con la clase `columns` permite al usuario especificar una separación entre las columnas (de forma predeterminada, si no se utiliza el atributo `colsep`, la separación se establece en 0).

Sintaxis para multiples columnas:

```
<div class="columns" colsep="Xcm">

<div class="column" width="Y%">
... Text of the first column goes here ...
</div>

<div class="column" width="Z%">
... Text of the second column goes here ...
</div>

...

<div class="column" width="K%">
... Text of the nth column goes here ...
</div>

</div>
```

Ejemplo:

```
<div class="columns" colsep="0.3cm">

**Main sport per country:**
```

```
<div class="column" width="25%">
The main sport practised in Spain is soccer
</div>

<div class="column" width="30%">
The main sport practised in the USA is baseball
</div>

<div class="column" width="25%">
The main sport practised in India is cricket
</div>

</div>
```

Saltos de página

Teaching Markdown permite al usuario insertar saltos de página en un documento mediante el siguiente fragmento de texto:

```
### {.pagebreak}
```

Elementos de línea adicionales (*underline*, *alert*, *color*)

Estas extensiones permiten resaltar de diferentes maneras un fragmento de texto encerrado en un contenedor genérico de línea. Si especificamos la clase `underline` en el elemento de línea, el texto completo (incluido entre paréntesis) aparecerá subrayado. De manera similar, si usamos la clase `alert`, el texto se mostrará en rojo (de la misma manera que con el comando de Beamer `\alert`). Finalmente, si agregamos `color=<color_specification>`, el texto se mostrará en el color especificado.

Sintaxis:

```
[... Highlighted markdown text goes here ...]{.underline}
[... Highlighted markdown text goes here ...]{.alert}
[... Highlighted markdown text goes here ...]{color=<color_specification>}
```

Ejemplo:

```
The computer can be divided into three main parts, according to
[Von Neumann]{color=red} architecture: [CPU]{.underline}, [Memory]{.underline}
and [I/O]{.underline}.
```

Modificadores para el control del tamaño de fuente

Esta extensión permite al usuario establecer el tamaño de fuente del texto ubicado en una región determinada. Para tal fin, se debe encerrar el texto en un contenedor genérico de bloque o de línea, y usar el atributo `fontsize` para indicar el tamaño de fuente. Deben usarse los especificadores de formato de LaTeX (desde tamaños más pequeños a los más grandes): `tiny`, `scriptsize`, `footnotesize`, `small`, `normallsize`, `large`, `Large`, `LARGE`, `huge` y `Huge`.

Sintaxis (elemento de línea):

```
[... Markdown text goes here ...]{fontsize=size}
```

Sintaxis (elemento de bloque):

```
<div fontsize=[size_specifier]>
... Markdown text goes here ...
</div>
```

Ejemplo:

```
[Section 1: Introduction to the C language]{fontsize=huge}
<div fontsize="normalsize">
In this Section we will introduce the main concepts of the C programming language.
</div>
```

Sintaxis para alineación de bloques y figuras

Esta extensión permite al usuario especificar la alineación de un fragmento de texto encerrado en un contenedor genérico de bloque, y utilizando el atributo `align`. El fragmento de texto puede centrarse (`center`), alinearse a la izquierda (`left`) o a la derecha (`right`).

Sintaxis:

```
<div align=aligment_type_spec>
... Markdown text block goes here ...
</div>
```

Ejemplo:

```
<div align="left">
Dear sir,
</div>

<div align="center">
The purpose of this letter is to inform you that ...
</div>

<div align="right">
Madrid, 2017.
</div>
```

También se puede utilizar el atributo `align` para establecer la alineación de figuras.

Bloques de beamer avanzados

El paquete `beamer` de LaTeX permite la creación de diapositivas. Una característica muy útil de este paquete es que permite al usuario incluir bloques en cada diapositiva. Los bloques Beamer son como pequeñas diapositivas, con título y contenido, que pueden incluirse dentro de otra diapositiva Beamer (también conocida como *frame*).

Pandoc permite transformar el código Markdown a Beamer, usando la opción `-t beamer` en la línea de comando. Por defecto, los encabezados de nivel 3 (es decir, "`### Título`") se

renderizan como bloques de beamer convencionales con un título. Lamentablemente, la variante de Markdown reconocida por Pandoc no soporta marcas especiales para crear otro tipo de bloques de Beamer, como los `alertblocks` o los `exampleblocks`, para advertencias y ejemplos, respectivamente.

Para mitigar este problema, *Teaching Markdown* permite especificar el tipo de bloque de Beamer mediante el uso de un contenedor de bloque genérico de la siguiente manera:

```
<div class="[block_type]" title="[title]">
... Markdown text block goes here ...
</div>
```

donde `[title]` codifica el título del bloque beamer (se pueden usar dos idiomas usando la sintaxis para variables Dual Pandoc), y `[tipo_bloque]` puede tener uno de los siguientes valores :

- `alertblock`, para la inclusión de advertencias en las diapositivas.
- `exampleblock`, para la inclusión de ejemplos en las diapositivas.
- `block`, para agregar bloques de beamer convencionales.
- `whiteblock`, para agregar bloques de beamer regulares con un fondo blanco.
- `console`, para agregar bloques beamer que emulan una ventana de terminal (para la inclusión de ejemplos con comandos).

Animaciones de Beamer

Beamer permite la inclusión de animaciones sencillas en las diapositivas. Por ejemplo, el comando `\pause` nos permite revelar de forma secuencial partes de una diapositiva. De manera similar, `\only<N>{text}` hace posible indicar en qué paso `N` del proceso de *renderización* de una diapositiva se hará visible un fragmento de texto dado.

Mientras que el comando `\pause` se puede usar tal cual en nuestras diapositivas Markdown (para transformarlo en beamer), pandoc no admite la inclusión del comando `\only<N> {text}`. *Teaching Markdown* hace posible la inclusión de elementos Markdown dentro de un bloque "`only`" o usando un contenedor genérico de línea. Básicamente, la sintaxis soportada es la siguiente:

```
<div only="[step-spec]">
... Markdown text block goes here ...
</div>

[...Markdown text goes here ...]{only=[step-spec]}
```

Ejemplo

```
[This will be displayed all the time in the current slide]{only="1-"}
[This will show up in the second step]{only="2"}
[This will show up in the third step]{only="3"}
```

Bibliografía

[1] J. Gruber, «Markdown: Syntax». https://daringfireball.net/projects/markdown/syntax, 2018.

[2] L. Phillips, «Technical Writing with Pandoc and Panflute», *Linux J.*, vol. 2017, n.º 281, sep.

2017.

[3] J. MacFarlane, «Pandoc: a universal document converter». https://pandoc.org, 2018.

[4] A. García y J. C. Sáez, «Official website of the Dual Markdown Framework». http://dualmarkdown.org, 2018.

[5] «Sublime Text». https://www.sublimetext.com/, 2018.

[6] J. C. Sáez y K. Olcoz, «Panbuild plugin for Sublime Text». https://github.com/jcSáezal/SublimeText-Panbuild, 2018.

[7] F. Igual y J. C. Sáez, «Sample Dual Markdown documents». https://github.com/dualmarkdown/examples, 2018.

[8] D. Chaver, F. Igual, y J. C. Sáez, «The Dual Markdown Framework: Getting Started». https://dualmarkdown.github.io/getting-started/, 2018.

[9] J. C. Sáez, «Panbuild». https://github.com/jcSáezal/panbuild, 2018.

[10] A. García, D. Chaver, F. Castro, y J. C. Sáez, «Installing the Dual Markdown Framework». https://dualmarkdown.github.io/installation/, 2018.

[11] filters, «Pandoc filters». https://pandoc.org/filters.html, 2018.

[12] S. Correia, «Pandoc Panflute». http://scorreia.com/software/panflute/, 2018.

[13] «Typora». https://typora.io/, 2018.

[14] «Pandoc Crossref». https://github.com/lierdakil/pandoc-crossref, 2018.

[15] J. MacFarlane y otros, «Manual de usuario de Pandoc». https://pandoc.org/MANUAL.html, 2018.

[16] J. C. Sáez y otros, «Pandoc filters for the Dual Markdown Framework». https://github.com/dualmarkdown, 2018.

[17] J. MacFarlane y otros, «YAML Metadata Blocks». https://pandoc.org/MANUAL.html#extension-yaml_metadata_block, 2018.

[18] D. Chaver, F. Castro, y J. C. Sáez, «Documentation on the Dual Markdown's extensions for Pandoc». https://dualmarkdown.github.io/documentation/, 2018.

[19] «NSIS». http://nsis.sourceforge.net/Main_Page, 2018.

[20] S. Sudre, «Packages». http://s.sudre.free.fr/Software/Packages/about.html, 2018.

[21] J. C. Sáez, «Documentación oficial de Panbuild». https://github.com/jcSáezal/panbuild/blob/master/README.md, 2018.

[22] «MiKTeX». https://miktex.org/, 2018.

[23] «MacTeX». https://www.tug.org/mactex/, 2018.

[24] «TeX Live». https://www.tug.org/texlive/, 2018.

[25] C. Demko, «Pandoc Numbering». https://github.com/chdemko/pandoc-numbering, 2018.