



## FACULTAD DE INFORMÁTICA

# TESINA DE LICENCIATURA

**TÍTULO:** Migración completa de artículos de Open Journal Systems

**AUTORES:** Maceri, Santiago

**DIRECTOR:** De Giusti, Marisa Raquel

**CODIRECTOR:-**

**ASESOR PROFESIONAL:** Villarreal, Gonzalo Luján

**CARRERA:** Licenciatura en Sistemas

### Resumen

*El presente trabajo propone un desarrollo que permita la importación y exportación de artículos entre plataformas OJS, incluyendo todas las etapas por las que estos pasan, los actores que intervienen en el flujo de trabajo, comentarios de revisores y del autor del artículo, archivos relacionados al artículo y sus revisiones por pares.*

### Palabras Clave

*Revistas científicas, portales de revistas, circuito editorial, revisión por pares, OJS, plugin, interoperabilidad, extensión, exportación, importación, XML*

### Conclusiones

*Se pudo concretar el trabajo planteado, logrando así una exportación más completa de los artículos entre OJS. Además, se expuso cómo es el proceso para extender el plugin de exportación, por lo que este trabajo deja abierta la posibilidad para futuros proyectos que tengan como objetivo desarrollar aún más sus funcionalidades.*

### Trabajos Realizados

*Se analizó el proceso editorial que implementan las revistas científicas para evaluar los artículos que quieren ser publicados en las mismas.*

*Se estudiaron las funcionalidades de OJS haciendo foco en el proceso editorial y las ventajas que brinda a los equipos editoriales.*

*Se analizó y extendió el actual plugin de OJS, Native Export/Import plugin para que, de ahora en más, la exportación de los artículos incluya todas las etapas por las que estos pasan, los actores que intervienen en el flujo de trabajo, comentarios de revisores y del autor del artículo, archivos relacionados al artículo y sus revisiones por pares.*

### Trabajos Futuros

*Dado que este trabajo está enfocado en una versión de OJS específica (3.2.1.1), se plantea un trabajo que tenga como objetivo la extensión del plugin en base a las distintas versiones de OJS. Además, se promueve la idea de crear un estándar de interoperabilidad entre sistemas de OJS, pudiendo así lograr exportaciones entre diferentes versiones.*

*Asimismo, se propone el desarrollo de un sistema de control de errores, ya que actualmente el plugin no posee uno.*

# Migración completa de artículos de Open Journal Systems

---

Autor: Maceri, Santiago

Directora: Dra. De Giusti, Marisa Raquel

Asesor: Dr. Villarreal, Gonzalo Luján

Facultad de Informática UNLP

Abril 2021

# Índice

<b>Capítulo 1 - Introducción</b>	<b>3</b>
Objetivo	3
Motivación	3
Organización y Metodología	6
<b>Capítulo 2 - Estado del arte</b>	<b>8</b>
Proceso editorial	9
Portales de revistas	12
OJS	13
Comunidad OJS	15
Implementación y ventajas de OJS	16
OJS en Argentina	17
OJS y las ventajas de sus plugins	18
<b>Capítulo 3 - Propuesta</b>	<b>21</b>
Posibles soluciones	21
Native export/import plugin	23
Alcance de Native Export/Import plugin	24
Desarrollo de la extensión	30
<b>Capítulo 4 - Pruebas de migraciones</b>	<b>42</b>
Prueba de campo	45
Caso 1: Envío en etapa de submission	46
Caso 2: Envío cancelado	47
Caso 3: Envío en revision	48
Caso 4: Envío en copyediting	50
Caso 5: Envío en production	51
Caso 6: Envío publicado	51
<b>Capítulo 5 - Conclusiones y Trabajos Futuros</b>	<b>55</b>
Ventajas	56
Desventajas	58
Trabajos a futuro	60
Conclusión	63
<b>Referencias</b>	<b>64</b>
<b>Anexo</b>	<b>67</b>
XML native.xsd actual de OJS	67
XML native.xsd extendido	72
XML con artículos	78
<b>Agradecimientos</b>	<b>84</b>

# Capítulo 1 - Introducción

## Objetivo

En este trabajo se propone un desarrollo de software que amplía las capacidades de importación y exportación de artículos del sistema de gestión de publicaciones periódicas *Open Journal Systems* (OJS). Se propone la ampliación del esquema original del formato XML *Native* de OJS y la adecuación de las herramientas de importación y exportación preexistentes para aprovechar el nuevo esquema presentado. La propuesta permite abarcar una mayor cantidad de metadatos de artículos, de números y de autores, e integra la información pública con la surgida de los procesos editoriales, lo que incluye etapas de edición y revisión, actores que intervinieron en cada etapa, versiones de artículos y discusiones e intercambios entre equipos editoriales, autores y revisores externos. Esta ampliación permite agilizar el proceso de migración de revistas desde distintas instalaciones de OJS.

## Motivación

La cantidad de instalaciones de OJS ha aumentado de manera sostenida en los últimos años. Editoriales privadas, organizaciones científicas, instituciones académicas y equipos editoriales tienen al alcance de su mano una herramienta muy completa para gestionar sus procesos editoriales, que está construida sobre tecnologías ya establecidas y de uso extendido, cuya licencia de código abierto permite su uso libre y con una extensa comunidad de usuarios y desarrolladores que brinda documentación, mejoras y soporte. El éxito de OJS ha generado que decenas de miles de revistas funcionen sobre alguna instalación del sistema, lo que ha provocado, con el correr del tiempo, la necesidad de migrar revistas entre diferentes instalaciones del software: revistas cuyo control pasa a manos de otras instituciones, organizaciones que buscan reorganizar o simplificar la gestión de sus portales web, e instituciones que se fusionan o que unifican sus portales por diversos motivos (menores costos de mantenimiento, vinculación entre empresas, unificación de equipos técnicos, entre otros), son algunas de las causas que provocan este movimiento de revistas entre instalaciones.

El caso de la UNLP, donde surge este proyecto, no es ajeno a esta realidad. Esta universidad cuenta con una gran cantidad de publicaciones periódicas propias que han surgido en distintos

momentos de su historia y bajo diferentes realidades: un equipo de científicos decide lanzar una publicación sobre su área de investigación, un departamento determina la sistematización de las publicaciones de sus equipos de docentes-investigadores, una dependencia comienza a brindar el servicio de gestión editorial y asesoramiento técnico, un convenio con una organización científica fomenta el lanzamiento de una publicación, son sólo algunos ejemplos que han provocado el surgimiento de revistas y de equipos de gestión de revistas en los últimos siglos<sup>1</sup>. Este surgimiento constante de nuevas publicaciones a lo largo de los años ha sido muy dispar: los procesos editoriales que aplicaba cada equipo se ajustaban en muchos casos a sus capacidades, o a los conocimientos y experiencias previas de los responsables de cada revista, y no al seguimiento de normas preestablecidas; la incorporación de herramientas tecnológicas, en los casos en los que contaban con alguna, se ajustaban a la disponibilidad de personal técnico y de fondos para contratar servicios de base tecnológica (por ejemplo, servidores, monitoreo y backups); y el sostenimiento y adecuación a lo largo del tiempo de los sistemas informáticos dependía en muchos casos de la buena voluntad de los miembros de los equipos editoriales, o de algún familiar o amigo con conocimientos técnicos para realizar el trabajo. Luego de mucho tiempo de iniciativas aisladas, distribuidas y desorganizadas, y sobre todo de falta de conocimiento e integración de información sobre publicaciones periódicas en la UNLP, en los últimos 20 años comenzó a aclararse el panorama a partir de diversas iniciativas que promovieron la centralización de información, la generación de servicios para promover la mejora continua y la incorporación sistemática de herramientas tecnológicas para todas las publicaciones periódicas de la institución. Entre estas iniciativas se destacan la creación del repositorio institucional SEDICI<sup>2</sup> (2003) y del repositorio de la Facultad de Humanidades y Ciencias de la Educación<sup>3</sup> (2007), la creación del Portal de Revistas de la UNLP<sup>4</sup> (2008) y el posterior surgimiento de portales de revistas en otras facultades (e.g., FAHCE<sup>5</sup>, Periodismo<sup>6</sup> y Bellas Artes<sup>7</sup>), las convocatorias para subsidios a revistas científicas de la UNLP (2012-2013) y,

---

<sup>1</sup> En rigor, existen revistas gestionadas desde la UNLP cuyos orígenes datan de varias décadas. Ejemplos de esto son la Revista del Museo de La Plata (1890) y la Revista de la Facultad de Agronomía (1895). Para más información, se aconseja explorar la colección de Revistas del repositorio SEDICI

<http://sedici.unlp.edu.ar/handle/10915/51>

<sup>2</sup> Servicio de Difusión de la Creación Intelectual es el Repositorio Institucional de la Universidad Nacional de La Plata <http://sedici.unlp.edu.ar>

<sup>3</sup> Memoria Académica, repositorio de la FAHCE <http://www.memoria.fahce.unlp.edu.ar>

<sup>4</sup> Portal de Revistas de la UNLP, Universidad Nacional de La Plata <https://revistas.unlp.edu.ar/index>

<sup>5</sup> Portal de Revistas de la FAHCE, Facultad de Humanidades y Ciencias de la Educación <https://www.revistas.fahce.unlp.edu.ar/>

<sup>6</sup> Portal de Revistas de Periodismo y Comunicación <https://perio.unlp.edu.ar/ojs/>

<sup>7</sup> Editorial Papel Cocido de la FBA, Facultad de Bellas Artes <http://papelcosido.fba.unlp.edu.ar/ojs/>

recientemente, la creación de la Coordinación de Revistas de la Universidad Nacional de La Plata (2018).

Bajo esta nueva realidad, donde la información comenzó a estar más clara y organizada, empezaron a evidenciarse las disparidades y deficiencias entre distintos grupos y revistas: software inadecuado, sistemas desactualizados, procesos editoriales obsoletos, falta de servicios de valor agregado, etcétera. En particular, uno de los problemas que fueron surgiendo y que motivaron este proyecto, está vinculado a la integración de revistas aisladas hacia portales consolidados; existen muchos casos de revistas de la UNLP que poseen sus propias instalaciones de OJS, pero que se encuentran desactualizadas, sin personal técnico para mantenerlos y actualizarlos y que no cuentan con los servicios que brindan los portales centralizados de la institución. Estas publicaciones periódicas requieren su migración hacia otra instalación de OJS (por ejemplo, hacia el Portal de Revistas de la UNLP), donde ya funcionan otras revistas científicas. Actualmente, OJS no dispone de un proceso capaz de realizar la migración completa de revistas entre portales; las soluciones implican una migración incompleta, tanto en cuanto a tipos de recursos (usuarios, artículos) como a información (metadatos) asociados a dichos recursos, y se requiere un importante esfuerzo humano para completar la información no migrada, revisar metadatos y corregir errores. En este contexto, muchas migraciones no se están realizando, o en caso de iniciarse pueden durar meses o años hasta que se completan. En consecuencia, se evidenció que hacía falta una solución informática que agilice este proceso, especialmente en lo vinculado con la migración de cada propuesta o envío (publicado o no publicado) y de toda la información surgida a partir de los mismos. Además, se observó interés en este desarrollo en la comunidad internacional de usuarios de OJS<sup>8</sup>, lo que fomentó la implementación de una solución más genérica, que sirva no sólo a la UNLP sino que pueda ser compartida con otras instituciones académicas y científicas que requieran agilizar el traslado de revistas entre instalaciones de OJS.

---

<sup>8</sup> Ejemplos de discusiones en diversos foros que tiene como objetivo promover la ampliación del plugin:  
<https://github.com/pkp/pkp-lib/issues/3261>  
<https://forum.pkp.sfu.ca/t/export-and-import-a-whole-journal/48749>

## Organización y Metodología

Este trabajo está organizado en varias secciones. En primer lugar, dado que aquí se mencionan muchos aspectos vinculados con las revistas científicas (metadatos, procesos editoriales, sistemas de revisión, etc.), se abordan algunos conceptos relacionados con la difusión científica y con la importancia que tienen las revistas en dicha difusión. Se hace hincapié en el concepto de calidad de una publicación científica, qué significa, por qué es importante y cómo se alcanza un nivel considerado de “buena calidad” para este tipo de revistas. Se enuncian aquí algunas de las formas más comunes con las que las instituciones gestionan sus revistas científicas y se realiza una introducción al software de OJS, su funcionamiento y sus principales componentes. El objetivo de esta breve introducción es unificar algunos conceptos centrales de este trabajo y brindar un panorama un poco más amplio para comprender mejor dónde encaja esta iniciativa en un marco institucional más amplio: la gestión y la visibilidad de publicaciones periódicas de la UNLP, la coordinación y el desarrollo de OJS por parte del Public Knowledge Project (PKP) y los sistemas internacionales de evaluación de revistas científicas.

En segundo lugar, se explica en detalle el desarrollo de este proyecto. Cuáles fueron las motivaciones que lo impulsaron, qué alternativas surgieron para plantear una solución, en qué consistió el desarrollo *per se*, qué problemáticas surgieron a lo largo del proyecto y qué resultados se obtuvieron con la nueva extensión que se realizó.

En tercer lugar, se plantea un entorno de pruebas que involucra dos instalaciones de OJS que tendrán incorporado el proyecto presentado en la sección anterior y donde se llevan a cabo diferentes migraciones que involucran distintos artículos con sus debidos procesos editoriales entre instalaciones. El propósito de este proceso es revisar el correcto funcionamiento del proyecto y resaltar las nuevas funcionalidades que este desarrollo incorpora a OJS. La metodología que se utilizó para la realización del desarrollo y las distintas pruebas consistió de un proceso iterativo de análisis, diseño e implementación de los distintos conjuntos de datos posibles de integrarse con el sistema de importación y exportación de OJS. En cada etapa de la iteración se evaluaba un nuevo requerimiento (por ejemplo, migración de revisiones de pares externos, o de discusiones entre equipos editoriales), se evaluaba cómo integrar el requerimiento con las herramientas de migración, se implementaba la extensión a las herramientas y se ponía a prueba para verificar que cumplía con los requerimientos planteados. Asimismo, se realizaron evaluaciones periódicas para verificar que las nuevas incorporaciones

de datos fueran compatibles con las incorporaciones en iteraciones anteriores. Finalmente, se realizaron pruebas integrales donde se evaluaron todas las incorporaciones en envíos en distintas etapas del proceso editorial, a fin de verificar que la información correspondiente a cada etapa era incorporada correctamente en el proceso de importación y exportación.

Por último, se realiza un análisis de las ventajas y desventajas que surgieron a partir de este desarrollo. Se plantean también algunas líneas de desarrollo futuro, que podrían resolver ciertos puntos mejorables de este trabajo, y se proponen también proyectos de ampliación para realizar migraciones de revistas aún más completas, o para generalizar estos procesos para casi cualquier versión de OJS.



## Capítulo 2 - Estado del arte

La comunicación y la difusión científica son los principales mecanismos que utilizan los miembros de la comunidad científica y académica para compartir con sus pares sus avances, sus modelos, sus resultados, sus proyectos, sus datos y todos los recursos que permiten comprender los avances, evaluarlos, replicar los experimentos y finalmente progresar hacia nuevos avances.

En la actualidad, el medio que más se utiliza como canal de comunicación es el de los artículos (*papers*) publicados en revistas científicas (*journals*). La Organización de las Naciones Unidas para la Educación la Ciencia y la Cultura (Jiménez y Castañeda, 2003) define a la revista científica como una:

[...] publicación periódica que presenta especialmente artículos científicos, escritos por autores diferentes, e información de actualidad sobre investigación y desarrollo de cualquier área de la ciencia. Tiene un nombre distintivo, se publica a intervalos regulares, por lo general varias veces al año, y cada entrega está numerada o fechada consecutivamente. Su componente básico, el artículo científico, es un escrito en prosa, de regular extensión, publicado como una contribución al progreso de una ciencia y arte.

Esta definición introduce algunos conceptos, muchos de los cuales han evolucionado considerablemente en los últimos años (un ejemplo de estos cambios es el mecanismo de publicación continua, cada vez más frecuente en muchas revistas científicas) y pone el foco sobre la idea de “publicación científica”. ¿Por qué “científica”? Como menciona Miyahira, en *El arbitraje editorial en las revistas médicas* (1995), las publicaciones científicas se caracterizan por estar respaldadas por cuerpos editoriales que mediante el proceso denominado arbitraje editorial, revisión por pares (*peer-review*) o revisión científica, “garantizan” estándares mínimos de calidad de lo que se publica en sus revistas. La definición presentada por este autor indica que la revisión por pares, durante todo el proceso editorial, es una parte fundamental en el proceso de publicación de artículos científicos.

Dentro de este marco, el objetivo principal de las publicaciones es comunicar e informar sobre su contenido, y para lograr dicho objetivo es necesario tener en cuenta que estas deben pasar por un proceso editorial, adaptado a los requerimientos y capacidades de cada equipo, pero que

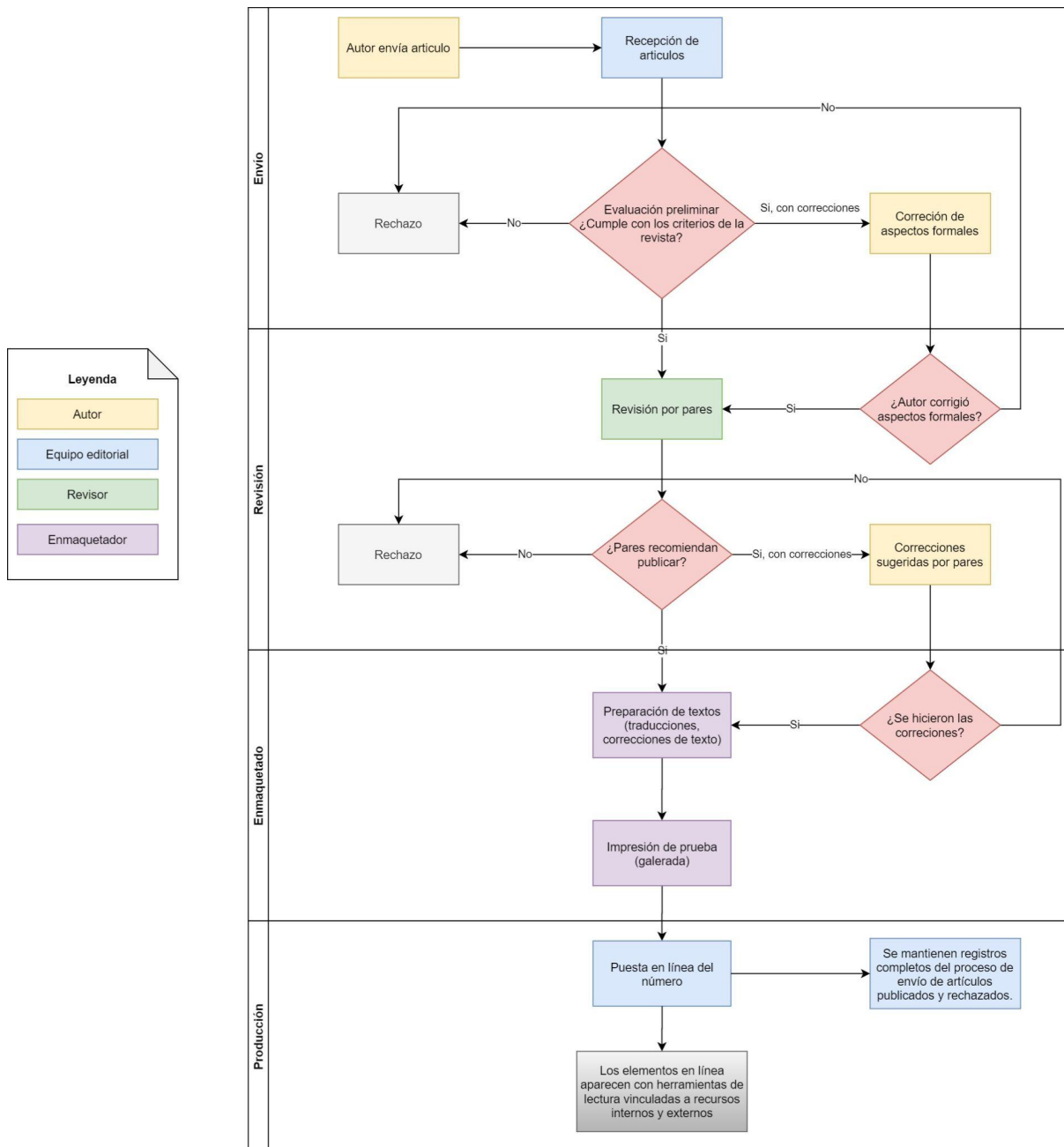
al fin y al cabo debe resultar en un producto de calidad, tanto editorial como científica (Rozemblum et al., 2015). Ahora bien, antes de hablar del proceso editorial, es necesario aclarar el concepto de calidad en una revista científica. Es importante conocer cómo perciben los usuarios la calidad de un servicio o producto, haciendo énfasis en la confiabilidad, en la prontitud de respuesta, en la seguridad, en la accesibilidad y en la empatía. Así también, se debe tener en cuenta que la satisfacción del lector de una publicación científica tiene relación directa con un proceso editorial y su producto de calidad, en concordancia con la satisfacción de sus necesidades. Un ejemplo de indicadores de calidad puede el catálogo de Latindex<sup>9</sup>, el que indica una serie de características que las revistas deben cumplir para postularse.

## Proceso editorial

Con estas características en consideración, es necesario determinar el flujo del proceso editorial. Para Rojas y Rivera (2011) el proceso editorial se inicia con la recepción de un artículo y avanza hacia la etapa de publicación de la revista. Además, los autores remarcan que este proceso implica, como mínimo, recibir colaboraciones, supervisar el proceso de arbitraje por pares y de revisión, dar formato a los trabajos para su almacenamiento, realizar el marcado para índices y depositarlos en un sitio web de acceso público y estable. Asimismo, como sugiere Rozemblum (2015), luego de la etapa de publicación de un artículo, los equipos editoriales realizan ciertas tareas necesarias para la difusión, la visibilidad y el alcance de los contenidos del mismo (como incorporación de metadatos en distintos idiomas, mención e indización en diferentes bases de datos como Latindex, SciELO, SCOPUS, etc.) Este tipo de tareas que promueven el factor visibilidad/difusión y facilitan el acceso de un artículo son de suma importancia. Esto es debido a que, para que el conocimiento científico genere un impacto, es esencial hacer visible los resultados para facilitar su consulta y uso para la comunidad.

---

<sup>9</sup> Características de calidad del Catálogo 2.0, Latindex <https://www.latindex.org/latindex/meto2>



Adaptación de Figura 2. Guía de Buenas Prácticas para Revistas Académicas de Acceso Abierto Rojas y Rivera (2011, p. 10)

Como indica Hernandez, en *Proceso editorial de una revista científica: cumpliendo con los requisitos de publicación* (2015), la recepción del artículo constituye el primer eslabón de la cadena de producción de una revista, pues los manuscritos deben tener una estructura definida en normas actualizadas según estándares que todo equipo editorial de una revista científica proponga. Rojas y Rivera (2011) denotan que en esta etapa, el editor puede hacer una evaluación general de la calidad y forma del documento recibido; aquí se produce un primer filtro que puede dejar

afuera artículos por temática o por una insuficiencia evidente de calidad, o bien se pueden pedir modificaciones por falta de apego a las normas de publicación. También es importante que el autor en este punto tenga información sobre las normas que rigen los derechos intelectuales del trabajo que está entregando a la revista.

Luego de la revisión por parte del editor, o revisión interna, comienza la etapa de revisión por pares (RPP). Según Berman et al. (2017), se habla de RPP cuando un par experto en la disciplina que desarrolla el autor, externo a la revista (o sea, que no pertenece al equipo editorial), independiente y libre de conflicto de intereses realiza la evaluación de un manuscrito, en un plazo estipulado. En su evaluación crítica y constructiva, el revisor debe valorar el artículo en cuanto a su aporte al conocimiento, a la calidad, al diseño, a la metodología, a la estructura, a la originalidad, a la validez de la discusión y de las conclusiones y, también, en la posibilidad de fraude. Además de sus comentarios, el revisor aporta recomendaciones al autor que permiten mejorar el manuscrito y aconseja al editor acerca de si este puede ser publicado, si requiere modificaciones o si debería rechazarse. Aquí vale destacar que, más allá de las actividades arriba mencionadas de los revisores, cada revista adapta el proceso de RPP y el rol de cada par según sus necesidades y sus posibilidades: en algunos casos el revisor solo evalúa el aporte a la investigación, o sólo intenta replicar los experimentos, o quizás analiza la metodología, la originalidad y la validez de la discusión. Dicho de otro modo: cada revista solicita a los revisores pares lo que cree conveniente y adecuado, según sus conocimientos, responsabilidad, tiempos disponibles, acceso a determinados instrumentos, etc. Un corolario interesante de esta diversidad en el contexto de este trabajo es que la etapa de RPP puede generar insumos muy diferentes en distintas revistas: documentos con notas del revisor, versiones comentadas del artículo, otros artículos similares, planillas con puntajes de distintos aspectos a considerar, entre otros.

Además, Berman et al. (2017) indican que existen diferentes métodos para realizar la RPP: la modalidad denominada doble ciego, caracterizada por el hecho de que los autores desconocen la identidad de los revisores, y viceversa, la modalidad de simple ciego, en que sólo los autores desconocen la identidad de los revisores, y la revisión abierta en que toda la comunidad científica está invitada a exponer sus críticas sobre *pre-prints* del artículo antes de ser sometido a un proceso editorial. También se han reportado experiencias con *post-print*, en que los artículos ya publicados son comentados por los lectores, por lo general (pero no necesariamente) miembros de la propia comunidad científica.

Con respecto a la descripción del proceso editorial, es necesario que los autores sean contactados en distintos puntos del proceso, por ejemplo para acusar recibo de su artículo, para indicar cuándo este ha sido enviado a revisión, para comunicar los resultados de la evaluación y pedir las correspondientes modificaciones al trabajo, o para informar del rechazo del artículo en caso de que no cumpla con los estándares de calidad de la revista. Según Hernandez (2015), el rechazo de un artículo no siempre se debe a que este no sea de calidad, sino que puede no ser adecuado a la temática de la revista. Se debe aclarar que un artículo rechazado no es necesariamente malo, sino simplemente uno al que no se le ha otorgado suficiente prioridad de publicación (Bosch et al., 2002). El rechazo también puede deberse a que el artículo no aporta mucho en el tema que está tratando, o a que es muy similar a otro publicado.

Dentro del proceso editorial es fundamental llevar un correcto seguimiento de la fase en que se encuentra cada artículo, de manera de ser capaz de manejar eficientemente los tiempos de la edición de cada número. Los autores requieren retroalimentación acerca de en qué estado de evaluación está su artículo (Rojas y Rivera, 2011). A lo largo del tiempo, la información que se genera en los distintos procesos de revisión de artículos también brinda un insumo muy importante para el equipo editorial que le permitirá mejorar sus procesos y le asistirá en la toma de decisiones: qué revisores responden en menor tiempo, con cuánta profundidad realizan la revisión de cada artículo, en qué áreas específicas de la ciencia se desenvuelven mejor. Esto se suma a la información que la revista recolecta a lo largo del tiempo sobre los revisores, como ser las instituciones a las que pertenecen o las empresas con las que colaboran, que les permite no sólo identificar a las personas más idóneas para revisar cada trabajo, sino también minimizar los conflictos de intereses a la hora de hacerlo (por ejemplo, cuando autores y revisores pertenecen al mismo grupo de investigación o a la misma institución). Existen incluso propuestas para automatizar la selección de los revisores pares, basadas en precisamente toda la información histórica recolectada (Houssos et al., 2011).

## Portales de revistas

La creación de portales de revistas que agrupen en un único sitio web un número importante de títulos es una de las tendencias del ámbito científico. Según indican Abadal y Rius Alcaraz (2008), se trata de una estrategia de difusión y posiblemente comercialización, que implica la agrupación de títulos para facilitar las búsquedas de los usuarios, los cuales pueden encontrar

en una única plataforma artículos de todas las revistas. Esto contribuye a la mejora de la difusión de los contenidos publicados por las revistas científicas universitarias. El portal típico es aquel que dispone de un amplio abanico de revistas digitales, ya sean especializadas en una temática o de carácter multidisciplinar, y que incorpora algún sistema de recuperación de la información que permite acceder a los contenidos a partir de búsquedas por autor, por título, por materias, por fechas, etc. Además, suelen incluir facilidades para gestionar las referencias, suscribirse al servicio de alertas, etc. Del mismo modo, los portales de revistas proporcionan comunicación entre los autores, los editores y los revisores, lo que es de gran importancia para llevar a cabo un circuito editorial transparente, y aportan guías de publicación que son utilizadas por los autores. Algunos modelos de referencia son los que nos ofrecen los editores comerciales, como ScienceDirect (Elsevier) o SpringerLink (Springer), y los de libre acceso e institucionales, como el Portal de Revistas de la UNLP, el Portal de revistas de la FAHCE o, el Portal de revistas de Periodismo.

Dadas las ventajas mencionadas por los portales de revistas, los investigadores y sus producciones podrían aumentar su visibilidad, participar activamente en la discusión científica, validar mejor sus resultados y aumentar la colaboración internacional. A mayor visibilidad, mejor validación de las investigaciones, mayor reconocimiento del investigador local y aumento en la colaboración con la producción de más ciencia (Polanco-Cortés y Garro Acón, 2013). Los portales de revistas gestionados desde las universidades públicas pueden ayudar a las publicaciones a mejorar su calidad sin que esto implique disminuir los esfuerzos por entrar en otros índices y catálogos, así como el mejoramiento constante de las prácticas editoriales y científicas.

Existen diversas maneras para llevar a cabo la creación y la gestión de portales de revistas, ya sean desarrollos propios, comercializados u *open source*. Dada la motivación y el contexto de desarrollo de en este trabajo, se pondrá el foco sobre OJS como sistema de gestión de publicaciones periódicas.

## OJS

OJS es una plataforma de software de código abierto utilizada para gestionar y difundir publicaciones periódicas, que pueden ser: revistas científicas y académicas y publicaciones institucionales o publicaciones de eventos. Originalmente desarrollada y lanzada por el Public

Knowledge Project (Proyecto para el Conocimiento Público, PKP) en 2001 para mejorar el acceso a la investigación. Según afirma el sitio web de PKP, OJS es la plataforma de publicación de revistas de código abierto más utilizada en la actualidad, con más de 10000 revistas que la utilizan en todo el mundo [PKP20].

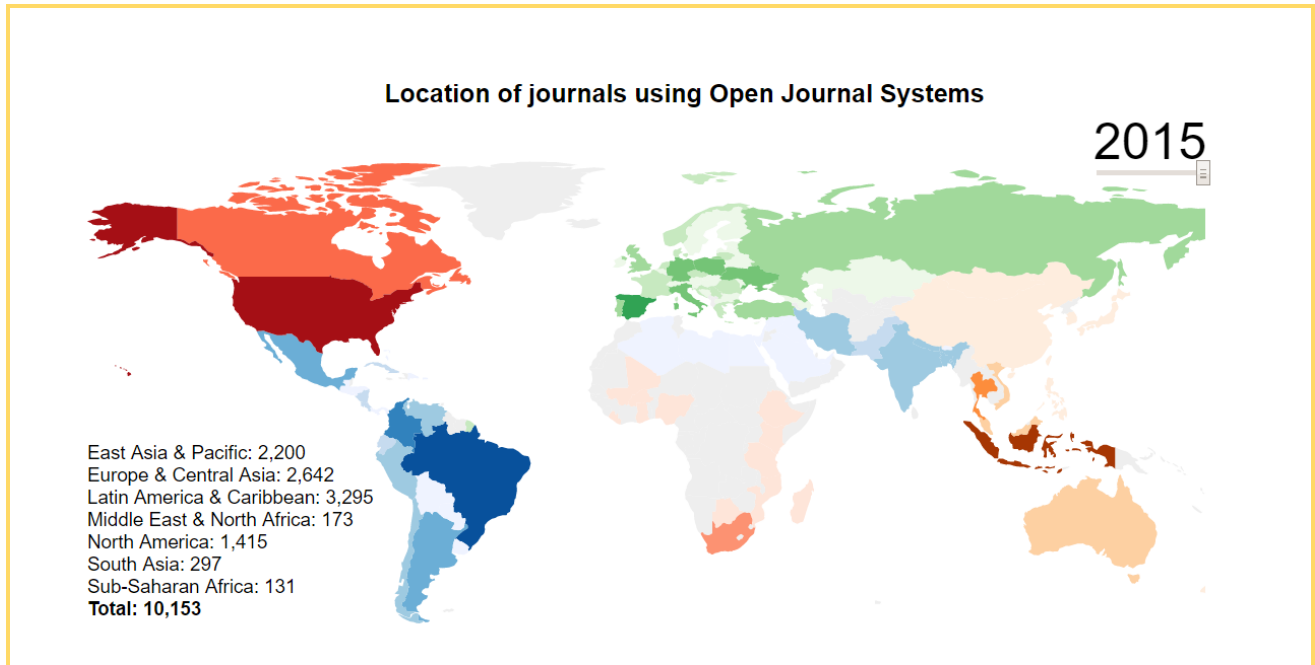


Gráfico 1. Revistas que utilizaban OJS en todo el mundo en el año 2015, organizadas por región/continente y por país. Fuente: <https://pkp.sfu.ca/2015/10/01/how-many-journals-use-ojs/>

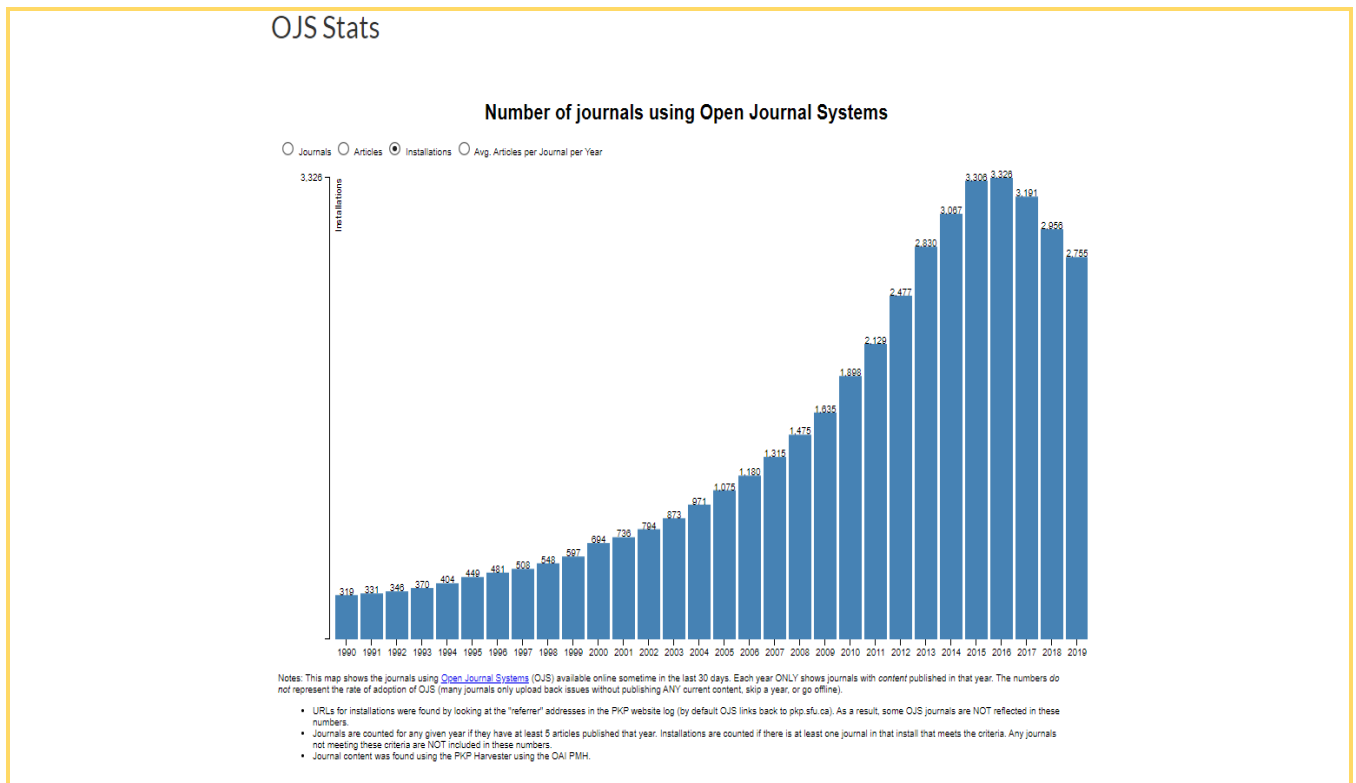


Gráfico 2. Gráfico interactivo que muestra la cantidad de revistas que utilizaron OJS a lo largo de los años.

Información extraída de <https://pkp.sfu.ca/2015/10/01/how-many-journals-use-ojs/>

## Comunidad OJS

Resulta interesante notar que OJS cuenta con un grupo de financiadores que, según sostienen los fundadores de PKP, sus actividades no serían posibles sin su apoyo continuo. Estos comparten el compromiso con el desarrollo y soporte a modelos alternativos de la publicación científica.

PKP también es uno de los servicios seleccionados como parte de la Coalición Global de Sostenibilidad para Servicios de Ciencia Abierta (SCOSS), junto con *Directory of Open Access Books (DOAB)*, *Open Access Publishing in European Networks (OAPEN)* y *OpenCitations*. Dentro del grupo de sus patrocinadores, se pueden destacar *French National Fund for Open Science (FNSO)*, *KU Leuven Bibliothekenbib*, *University of Toronto Libraries*, *SciELO*, entre otros. Además, PKP mantiene fuertes lazos y cuenta con el apoyo de otras instituciones comprometidas con modelos alternativos de publicación académica. Estas instituciones actúan como socios de desarrollo de PKP a fin de garantizar la mejora de software de código abierto y la calidad de las



publicaciones académicas. Entre ellos se pueden destacar Ontario Council of University Libraries (OCUL), Stanford University, Simon Fraser University (SFU).

Por un lado, PKP ha construido una comunidad a lo largo de todo el mundo, compuesta por diversas instituciones. Sus comités y grupos de interés son una parte integral, brindándoles comentarios y asesoramiento a la comunidad en su conjunto.

Por otro lado, en muchos aspectos, desde el desarrollo, las traducciones y hasta la documentación, PKP depende en gran parte de su comunidad de acceso abierto, por lo que facilita un espacio a la comunidad para que aporte en sus foros de discusión, contribuya con desarrollo de software y de plugins, reporte errores y sugerencias a nuevas funcionalidades, comente con traducciones o documentación, cree o mejore manuales de uso del software de PKP, etc.

## Implementación y ventajas de OJS

En su trabajo, Torres et al. (2019), dan a entender que implementar este software ayuda a modernizar el proceso editorial de las actuales publicaciones científicas y apoya a la creación de otras de alto impacto.

OJS es un sistema de administración y publicación de revistas en línea que tiene un impacto muy alto. Por una parte, este sistema brinda un manejo eficiente y unificado para el proceso de los flujos editoriales que permite pensar en la implementación y ejecución del proyecto. Además, agiliza todas las actividades en el proceso de revisiones de artículos, con lo que se busca acelerar el acceso en la difusión de contenidos e investigaciones producidos por una revista, y cuenta con la capacidad para trabajar con documentos en distintos formatos de extensión, como Word, PDF, HTML, XML, etc.

Por otra parte, gracias a su licencia libre, su funcionalidad y la gran comunidad de desarrolladores que están detrás de este proyecto, OJS permite que la instalación, configuración y administración de sistemas de publicación y de gestión de revistas electrónicas se realice en el menor tiempo posible y con el mínimo esfuerzo, en comparación con un desarrollo web desde cero.

Open Journal System fue diseñado para facilitar los procesos de desarrollo de publicaciones en acceso abierto, revisadas por pares, administrando la infraestructura técnica no sólo para la presentación en línea de artículos de revistas, sino también para agilizar el flujo editorial,

incluyendo los envíos de artículos, múltiples números de rondas de revisión por pares e indexaciones. Esta herramienta permite la publicación de los contenidos en acceso abierto, fundamentalmente de las revistas científicas publicadas en línea, cuya flexibilidad y versatilidad le permite adaptarse a las necesidades y requerimientos de cada publicación (Aretio, 2014).

La implementación de este software logra que las revistas puedan posicionarse en los índices y en las bases de datos de manera más transparente y estandarizada, ya que sistemas como Latindex Catálogo 2.0, Erih Plus, DOAJ, Dialnet, MIAR, REDIB, Google académico y SCIELO interactúan de manera amigable con OJS. Cabe resaltar también que estos aspectos de normalización de características, que debe cumplir una revista científica de calidad y que ofrece OJS, han motivado a la digitalización rápida de muchas publicaciones científicas.

## OJS en Argentina

En Argentina, OJS es utilizado en un gran número de universidades públicas, entre las que se incluyen UNCUYO<sup>10</sup>, UNSa<sup>11</sup>, Rosario<sup>12</sup>, Córdoba<sup>13</sup>, Litoral<sup>14</sup>, entre otras. El uso de dicha plataforma también se extiende a universidades privadas (e.g., Universidad Austral<sup>15</sup>) y a revistas gestionadas por editoriales privadas o sociedades científicas, como es el caso de la Sociedad Argentina de Dermatología<sup>16</sup> (sello editorial Lugones) o la Asociación Paleontológica Argentina (revistas PEAPA<sup>17</sup> y Ameghiniana<sup>18</sup>), por citar algunos ejemplos. Como ya se indicó, se destaca también que la UNLP cuenta con varias instalaciones de OJS, entre las que se incluyen el Portal de Revistas de la UNLP, gestionado desde PREBI-SEDICI, Revistas de la FAHCE, Portal de Revistas de Periodismo y Comunicación, Editorial Papel Cosido, de la Facultad de Artes, y las Revistas TEyET<sup>19</sup> y JCS&T<sup>20</sup> de la Facultad de Informática.

---

<sup>10</sup> Portal de Revistas de la UNCUYO, Universidad Nacional de Cuyo <http://revistas.uncu.edu.ar/>

<sup>11</sup> Portal de Revistas de la UNSa, Universidad Nacional de Salta  
<http://portalderevistas.unsa.edu.ar/ojs/index.php/index>

<sup>12</sup> Portal de Revistas de la UNR, Universidad Nacional de Rosario <https://revistas.unr.edu.ar/>

<sup>13</sup> Portal de Revistas de la UNC, Universidad Nacional de Córdoba <https://revistas.unc.edu.ar/>

<sup>14</sup> Portal de Revistas de la UNL, Universidad Nacional del Litoral  
<https://bibliotecavirtual.unl.edu.ar/publicaciones/>

<sup>15</sup> Portal de Revistas de la Universidad Austral <https://ojs.austral.edu.ar/>

<sup>16</sup> Portal de Revistas de la Sociedad Argentina de Dermatología <http://www.dermatolarg.org.ar/>

<sup>17</sup> Publicación Electrónica de la Asociación Paleontológica Argentina <https://www.peapaleontologica.org.ar>

<sup>18</sup> Revista Ameghiniana de la Asociación Paleontológica Argentina <https://www.ameghiniana.org.ar>

<sup>19</sup> Revista TEyET, Revista Iberoamericana de Tecnología en Educación y Educación Tecnológica  
<https://tevet-revista.info.unlp.edu.ar/>

<sup>20</sup> Revista JCS&T, Journal of Computer Science and Technology <https://journal.info.unlp.edu.ar/>

Como se mencionó en el párrafo anterior, el portal de revistas de la UNSa fue uno de los tantos casos nombrados de universidades que utilizan OJS. En este sentido, Guzmán González y Díaz (2019) mencionan que los beneficios que OJS ofreció a la edición de las revistas científicas fueron muchos, pero principalmente se destaca la posibilidad de unificar y facilitar el proceso editorial en su complejidad. El flujo de información y de comunicación se torna más eficaz, eficiente y se reducen los tiempos. González y Díaz (2019) afirman que la plataforma facilita el trabajo que muchos investigadores vienen haciendo para dar a conocer sus investigaciones y las de sus colegas. Además, destacan que la creación del Portal de Revistas Científicas de la UNSa visibilizó aún más las publicaciones de las diferentes facultades al estar alojadas en una plataforma que cumple con el protocolo OAI-PMH (*Open Archives Initiative-Protocol for Metadata Harvesting*), el cual garantiza que cada uno de los datos puedan ser cosechados por otros sitios especializados, lo que permite la interoperabilidad de los sistemas.

## OJS y las ventajas de sus plugins

La administración de OJS ofrece una sección de *plugins* (también conocidos como extensiones/módulos). Estos plugins sirven para añadir aún más funcionalidad, como por ejemplo operar con más metadatos, estilos bibliográficos, exportaciones, identificadores persistentes (e.g., DOI), entre otras funciones. Cabe destacar que estos plugins pueden habilitarse o deshabilitarse (por el Journal Manager o algún usuario con los permisos necesarios), en función de las necesidades de cada revista.

Si bien OJS provee de manera predeterminada una gran cantidad de plugins, además proporciona una estructura lógica que simplifica la creación de nuevos, que es acompañada de documentación en línea que determina una serie de pautas a respetar para el desarrollo e instalación de estas extensiones. OJS establece ciertas categorías de plugins para que el usuario que desea crear su propia extensión la ubique adecuadamente donde pertenece. Algunas de estas categorías pueden ser:

- **Plugins de bloques (*Block plugins*):** visibilizan contenido que se puede mostrar en la barra lateral de cualquier página.

- **Plugins de Importación/Exportación (*Import/Export plugins*):** proporcionan herramientas para obtener datos dentro y fuera de OJS, y funcionalidad para migrar usuarios, envíos, ediciones anteriores y más.
- **Plugins de reportes (*Report plugins*):** brindan acceso a la descarga de un archivo CSV. El informe puede incluir detalles sobre estadísticas de uso de artículos, revisores, entre otras.
- **Plugins de temas (*Theme plugins*):** controlan el diseño y la maquetación de una revista o sitio web de prensa.

Estas son las categorías de plugins más comunes que destaca la documentación de PKP, sin embargo, existen otras como:

- **Plugins de autorización (*Auth plugins*):** permiten autorizar y sincronizar cuentas de usuario con una fuente de terceros.
- **Plugins de puerta de enlace (*Gateways plugins*):** permiten agregar una nueva URL y responder a las solicitudes a esa URL.
- **Plugins de metadatos (*Metadata plugins*):** permiten implementar una descripción de un formato de metadatos.
- **Plugin de formato de metadatos OAI (*oaiMetadataFormats plugins*):** agrega un formato de metadatos al endpoint OAI de la aplicación.
- **Plugins de métodos de pago (*Paymethod plugins*):** permiten implementar su propio manejo de pagos cuando utiliza suscripciones y tarifas de artículos.
- **Plugin de identificación de publicaciones (*publds plugins*):** permiten agregar soporte para identificadores de publicación como DOI.

La creación de plugins es la herramienta que otorga la posibilidad de desarrollo de extensiones personalizadas y, dado que OJS es un proyecto *open source*, todos estos desarrollos pueden ser compartidos con la comunidad que utiliza OJS y así promover un aporte a todas las instituciones que utilicen este software.

El concepto de plugins en OJS es de gran relevancia, ya que son herramientas útiles para obtener/mostrar/manipular información y pueden usarse a favor para realizar extensiones que permiten generar importantes resultados. Entre las distintas categorías de plugins que ofrece OJS se encuentra la de exportación e importación; dentro de esta existe, particularmente, *Native Import/Export Plugin*, que tiene como objetivo lograr una exportación e importación de

artículos. Actualmente, el proceso de migración de los artículos no contempla toda la información del flujo de trabajo que se lleva a cabo para cada uno, como quiénes intervienen en cada etapa, los comentarios que se efectuaron entre los revisores y el autor, y todo el proceso de revisión por pares (cada uno de los siguientes puntos será abarcado en el próximo capítulo). La información del proceso de revisión por pares es de suma importancia ya que abarca, básicamente, todo el análisis y trabajo que se le dedica a cada artículo para corroborar la calidad científica de sus contenidos y, luego, pueda ser publicado (o rechazado si corresponde). Esta información abarca tiempos de revisión, desempeño de los revisores (tiempos de revisión de cada revisor y rating del mismo), intercambio de ideas entre los revisores y el autor, etc. En síntesis, todo este proceso editorial al que se somete al artículo representa toda la historia del mismo y cómo el equipo editorial de la revista trabajó alrededor de él. Para una revista, llevar un control de cómo se realizan los procesos editoriales es fundamental, debido a que representa en gran parte la calidad de la revista, como se mencionó al comienzo de este capítulo. El factor calidad científica para una publicación periódica es de gran importancia, ya que influye en su impacto en la comunidad científica, en índices de búsqueda, a nivel comercial, legitimidad, etc. Por lo tanto, si una institución quiere migrar una revista, no puede perder toda la información que abarca el proceso editorial.

En los próximos capítulos, se hará énfasis en el proceso de migración actual, se dejará en evidencia la falta de información que este presenta (se mostrará por qué esta información es tan importante) y se propondrá una solución puntual para generar un proceso de información más complejo y más abarcativo.

## Capítulo 3 - Propuesta

Como se mencionó al final del capítulo anterior, mantener toda la información del proceso editorial de un artículo es de gran importancia y no existe una herramienta en OJS que contemple la migración completa del mismo. En este capítulo se hará énfasis en el desarrollo de una solución para esta problemática, se expondrán todas las alternativas que se tuvieron en cuenta para proponer dicha solución y se explicará el desarrollo propuesto por la tesis, su objetivo, estructura y funcionamiento.

### Posibles soluciones

Para poder llevar a cabo una mejora en el proceso de exportación, fue necesario realizar una investigación acerca de cómo obtener todos los metadatos de un envío de una instalación OJS, entender cuáles eran las dificultades que surgieron al incorporarlos y cómo lograr una importación exitosa en otra instalación OJS.

Una de las primeras alternativas que se encontraron en el proyecto fue el uso del protocolo OAI-PMH para la recolección de metadatos de los envíos. OAI-PMH utiliza transacciones HTTP para emitir preguntas y obtener respuestas entre un servidor o archivo y un cliente o servicio recolector de metadatos. Este servicio recolector puede pedir al servidor que le envíe metadatos según determinados criterios. En respuesta, el servidor devuelve un conjunto de registros en formato XML, incluyendo identificadores (URLs por ejemplo) de los objetos descritos en cada registro (Barrueco y Subirats-Coll, 2003). El PKP OAI Harvester permite crear un índice de búsqueda de los metadatos de archivos que cumplen con la iniciativa (OAI), como sitios que utilizan OJS. La mayoría de las revistas que actualmente utilizan OJS ahora pueden ser recolectadas por proveedores de servicio OAI (Deka, 2007). Si bien esta opción sigue un legítimo e impuesto estándar, los datos que se podían recolectar del envío eran solamente datos públicos, como el autor, su DOI, el artículo, número, etc. Sin embargo, estos datos resultaban insuficientes para generar el tipo de exportación que se buscaba. A esta problemática se suma que a través del harvesting OAI-PMH solamente se pueden obtener artículos que están publicados en portales, lo cual deja de lado a todos aquellos artículos que no fueron publicados, como por ejemplo los que están en un proceso de revisión y también los artículos que fueron rechazados. Por estas observaciones, esta primera alternativa no se consideró idónea.

Otra opción a considerar fue la obtención de los datos de los envíos a través de la API que provee DOAJ. *Directory of Open Access Journal* (DOAJ) es un directorio en línea que indexa y brinda acceso a revistas revisadas por pares, de acceso abierto y de alta calidad (Ahmar et al., 2018). Uno de los requerimientos que exige DOAJ es que el artículo que se desea incluir a su base de datos debe ser de acceso abierto por lo que un gran número de publicaciones que se encuentran en OJS son indexadas en DOAJ. La API de DOAJ no cuenta con una gran extensión pero provee endpoints para artículos, revistas, aplicaciones y ciertas búsquedas. Más específicamente, existe un endpoint puntual que, dado un id de un artículo facilita la obtención de información del mismo. A pesar de ello, sucede algo parecido que en la alternativa OAI PMH ya que la información que provee el endpoint de DOAJ solo está relacionada al artículo por lo que no hay información de todo el proceso de envío del mismo. Además, como también sucede en la primera alternativa, en la base de datos de DOAJ se encuentran solamente los artículos que han sido publicados, por lo que deja de lado a todos aquellos artículos que están en proceso de envío o fueron rechazados.

Otra alternativa considerada fue generar una exportación directa a través de scripts con sentencias SQL directamente a la base de datos del sistema OJS. Si bien es un acercamiento más directo y simple, requiere de una extensa comprensión de cómo está conformado el esquema completo de OJS y el proceso de desarrollo podría llegar a tornarse lento y muy propenso a errores. Esto se debe a que al no utilizar las herramientas planteadas por OJS para la toma de información, como por ejemplo el uso de los DAO creados por PKP, resultaría en una alta probabilidad que muchos de los controles/reglas de negocio planteados/as por los mismos se pasen por alto. Sumado a esta problemática, hay que tener en cuenta que en la base de datos no se encuentran los archivos de los envíos, por lo que habría que relacionar cada envío migrado con todos los archivos en cada de las etapas correspondientes.

Por último, surge una alternativa a partir del existente plugin *Native*, encargado de generar exportaciones de artículos entre diferentes OJS. Este plugin, plantea una exportación/importación a través de ficheros XML, similar al protocolo OAI PMH. Además existe ya un trabajo realizado por el equipo de trabajo PREBI-SEDICI (De Giusti et al., 2018), el cual expone un desarrollo que logró la importación de artículos OJS provenientes de archivos CSV que son procesados para generar archivos XML, los cuales son importados con el mismo plugin. Este desarrollo ofrece una primera aproximación a la solución buscada ya que plantea una exportación basada en archivos XML. Por lo tanto, se analizó la posibilidad de realizar una

extensión del proceso del plugin para luego generar una exportación más completa. Además, debido a la integración del plugin en OJS, se puede acceder a todo tipo de datos que posee el artículo, por lo que la falta de información ya no resultó ser una problemática. En consecuencia, se optó por esta última opción para el desarrollo del presente proyecto.

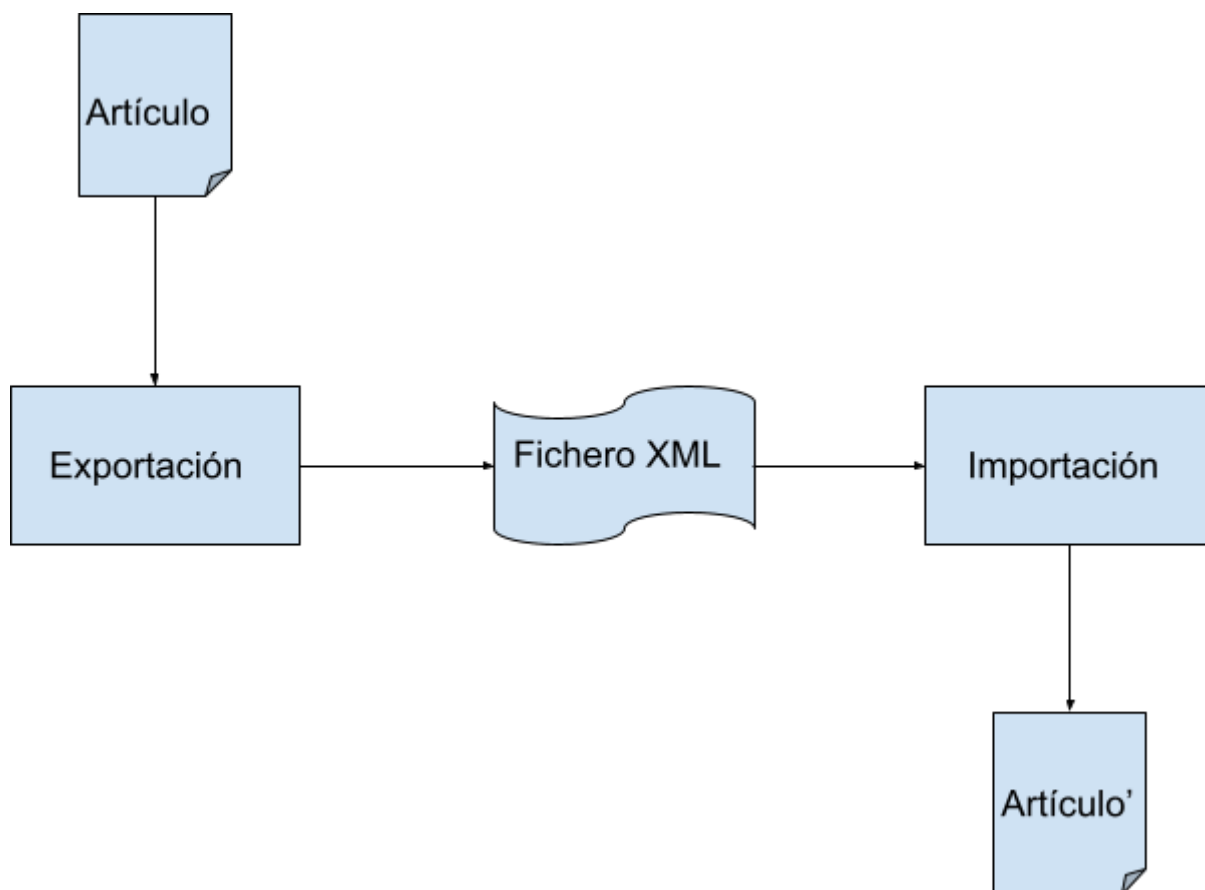
### *Native export/import plugin*

*Native Export/Import plugin* es una extensión desarrollada por el grupo PKP cuyo objetivo es la exportación e importación de artículos y ediciones entre distintas revistas. Cabe destacar que en este trabajo solo se hará foco en los artículos y, cuando se habla de éstos, no se hace referencia exclusivamente al archivo que contiene lo escrito por un autor en sí, sino a todo el proceso editorial realizado en el sistema. A todo este conjunto de información, OJS lo define como envío (o *submission*)

En sí, la funcionalidad del plugin es simple y se puede dividir en 2 secciones:

- Una sección de software que genera ficheros XML cuyo contenido incluye los artículos que se quieran exportar (sección de exportación)
- Una sección de software que procesa esos ficheros XML y guarda todos los datos que contiene en la revista, lo que genera una réplica del artículo original (sección de importación)





*Imagen 1: Diagrama de flujo que representa el proceso de exportación e importación de artículos en OJS. Cada artículo es representado mediante un formato XML, que es independiente de la representación subyacente de los datos (ej. mediante MySQL, MariaDB o PostgreSQL) y que incluye tanto los metadatos de los artículos como los archivos binarios. Fuente: elaboración propia.*

### ***Alcance de Native Export/Import plugin***

Como se mencionó anteriormente, esta extensión permite la exportación e importación (para simplificar, a partir de este momento se referirá al proceso como exportación) de artículos entre diferentes revistas. Más adelante se evaluará la completitud de este proceso, por lo tanto, un parámetro a tener en cuenta es la cantidad de información que aborda el mismo. Pero antes de cubrir el tema de completitud es necesario entender qué información se almacena en OJS relativa a un artículo y su proceso editorial. Por medio de las siguientes imágenes se mostrará cada parte que constituye un artículo y su proceso editorial en OJS:

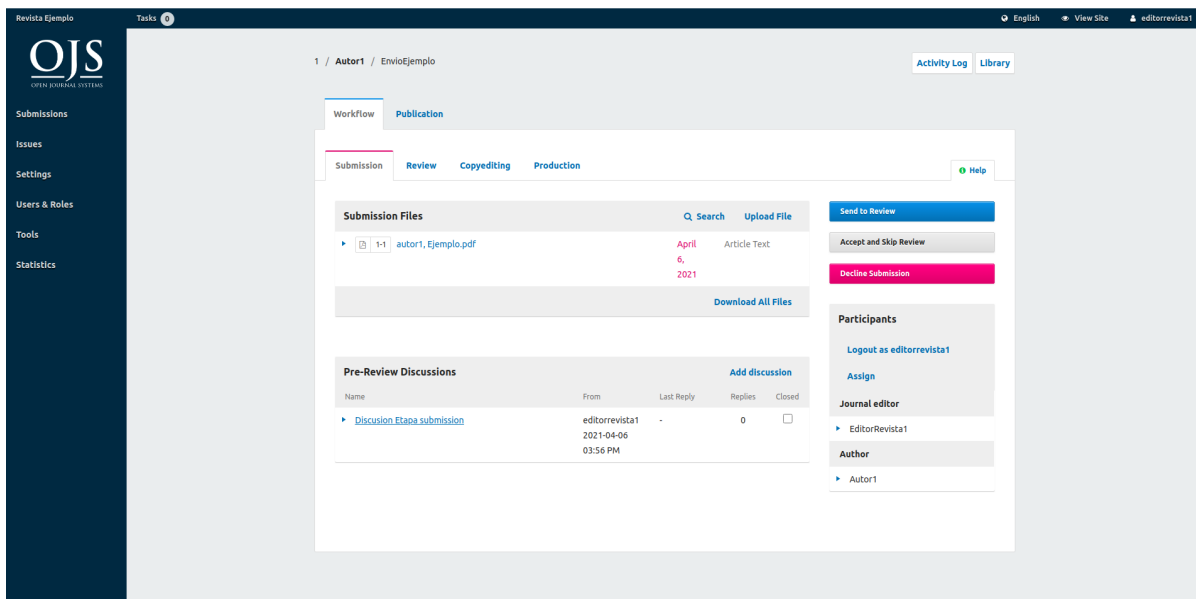


Imagen 2: Envío en la etapa de Envío Pendiente (submission). En esta etapa sólo se incluyen los archivos enviados por los autores. En caso que un editor haya sido asignado como participante, podrían también encontrarse intercambios (sección pre-review discussions) entre el editor y los autores. Fuente: extracción de la aplicación OJS, flujo de trabajo editorial.

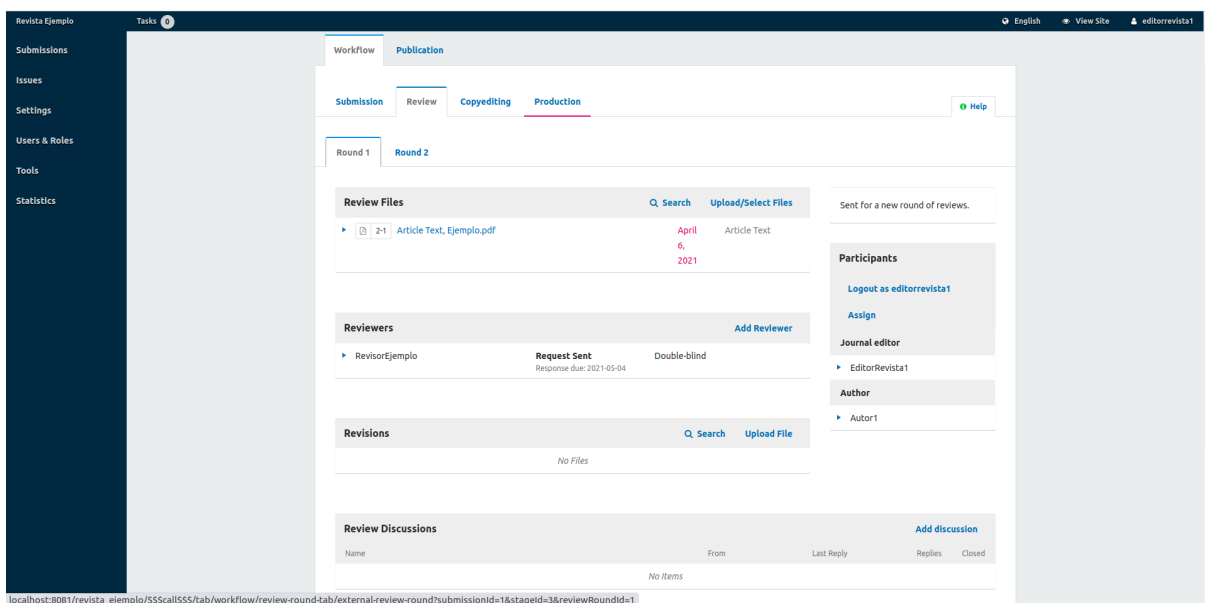


Imagen 3: Envío en etapa de Revisión (revision). En esta etapa, el artículo es sometido a una revisión por pares. Se indican que cambios debe realizar el autor a su artículo original para continuar con el proceso editorial. Fuente: extracción de la aplicación OJS, flujo de trabajo editorial.

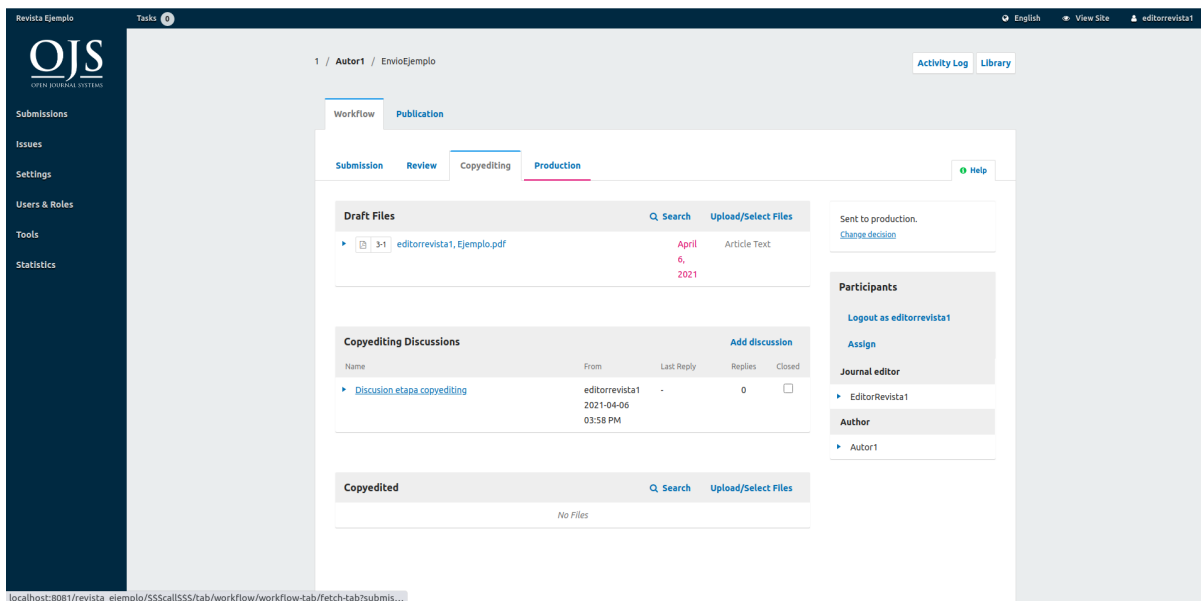


Imagen 4: Envío en etapa editorial (copyediting). En esta etapa el artículo es mejorado gracias al trabajo de un copyeditor. Los autores pueden tener la oportunidad de revisar las correcciones. Fuente: extracción de la aplicación OJS, flujo de trabajo editorial.

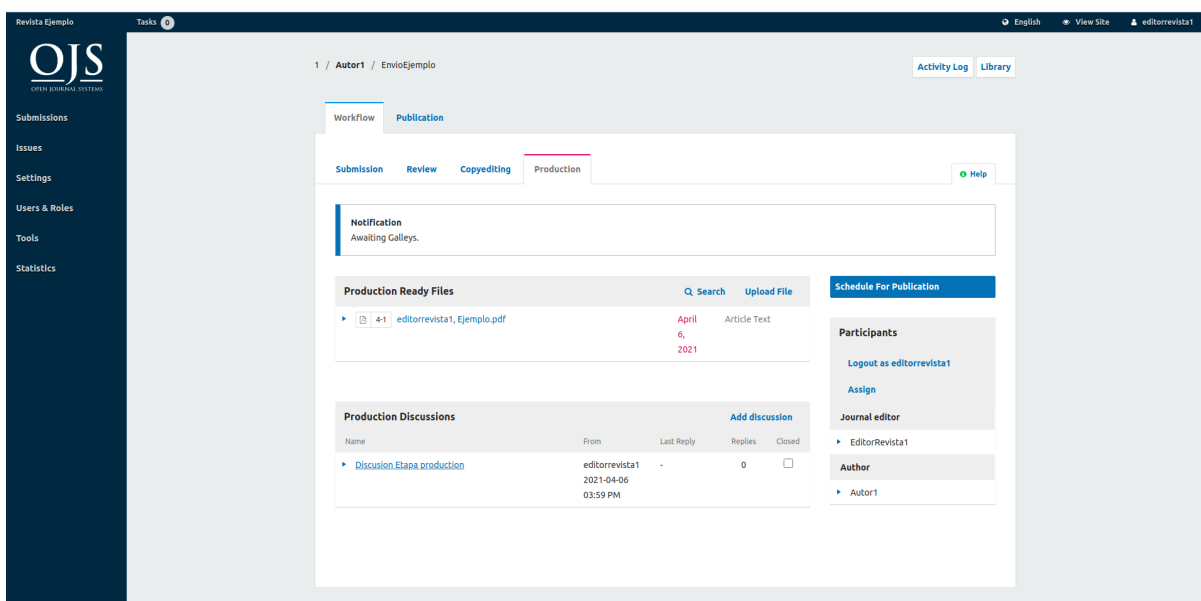
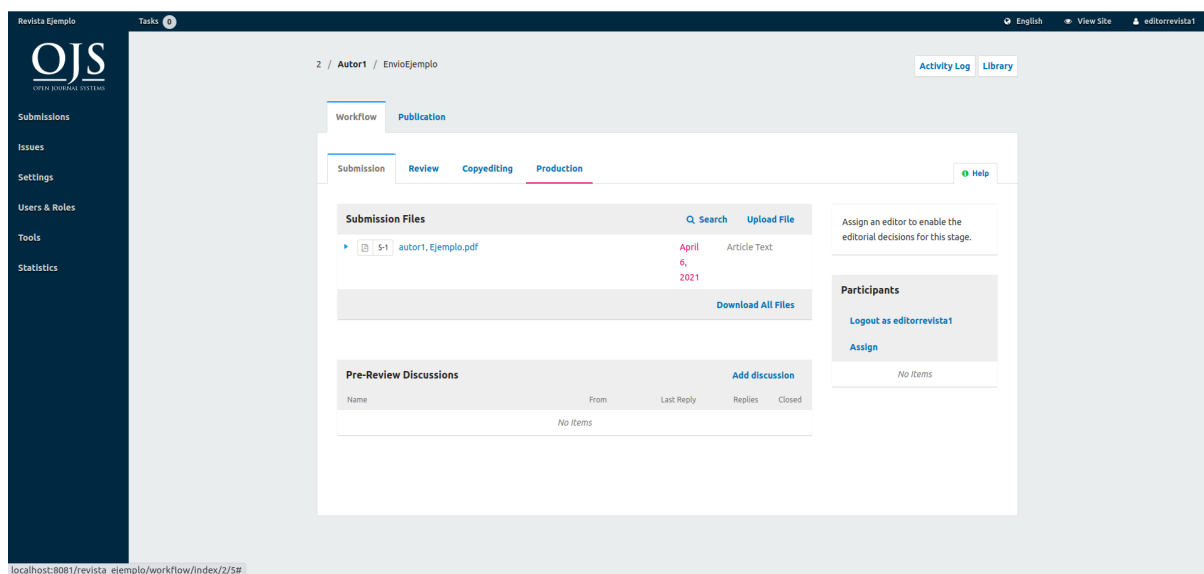


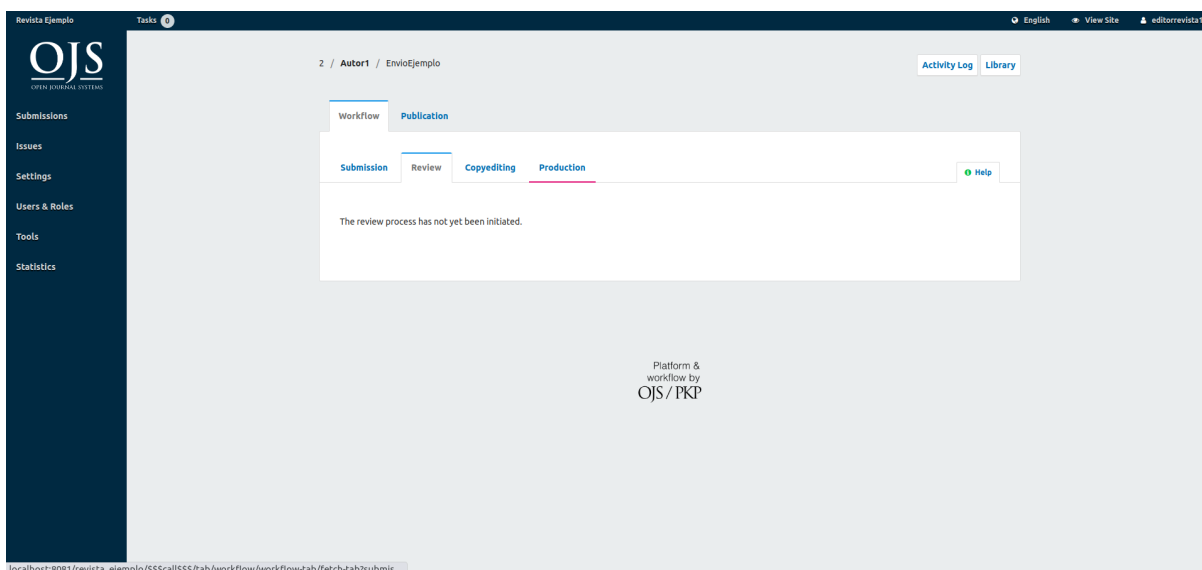
Imagen 5: Artículo en etapa de producción (production). En esta etapa, los archivos corregidos se convierten en galeras: HTML, XML, PDF, etc. y el autor tiene la oportunidad de revisarlas. Una vez que todos los participantes estén satisfechos, la presentación está programada para su publicación en un número futuro. Fuente: extracción de la aplicación OJS, flujo de trabajo editorial.

Este ejemplo es un caso típico de un envío que pasa por diversas etapas del proceso editorial antes de ser aceptado o rechazado. En cada etapa, el artículo es sometido a diversas acciones de los usuarios que intervienen en el proceso editorial, análisis del artículo, revisiones, correcciones, discusiones, etc. lo que genera una gran cantidad de metadatos que representan un historial de todo el proceso. Estos historiales son de gran valor, ya que, naturalmente, en caso de que se quiera revisar el proceso editorial de un artículo, se puede obtener a través de estos. Conjuntamente, estos metadatos pueden ser utilizados para generar diversos tipos de reportes estadísticos para diferentes tipos de evaluaciones, como por ejemplo tomar tiempos promedios de envíos, tiempos de respuesta de los autores, los revisores y los editores, evaluar ratings de estos en sus diferentes trabajos de revisión, evaluar motivos de rechazo o aceptación de artículos, etc.

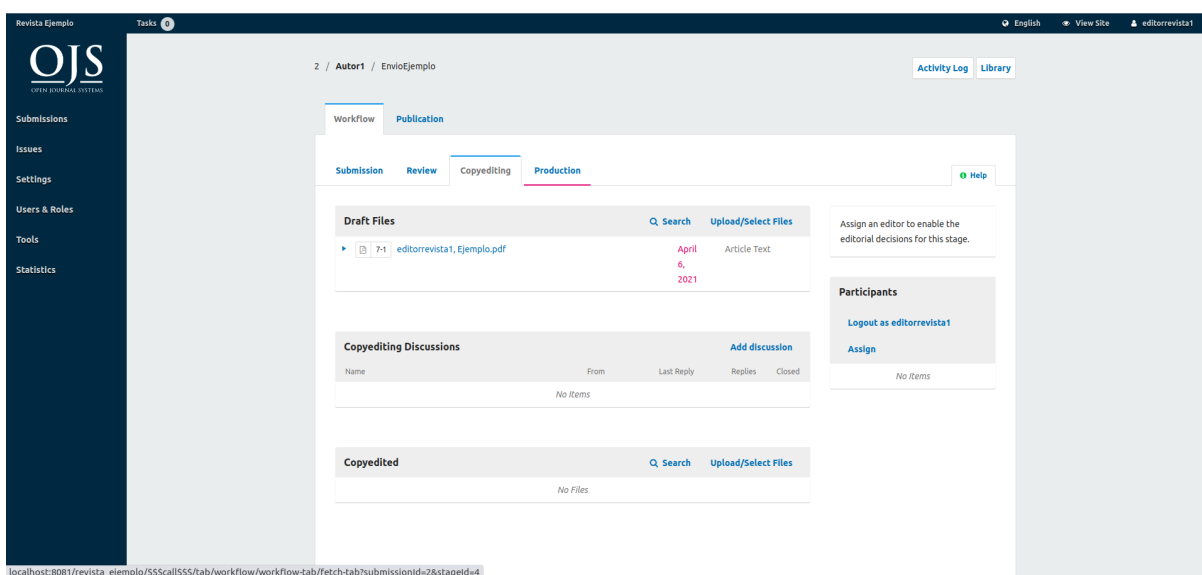
Si se toma como ejemplo este caso típico y se utiliza el actual plugin *Native* para exportarlo e importarlo podemos ver el siguiente resultado:



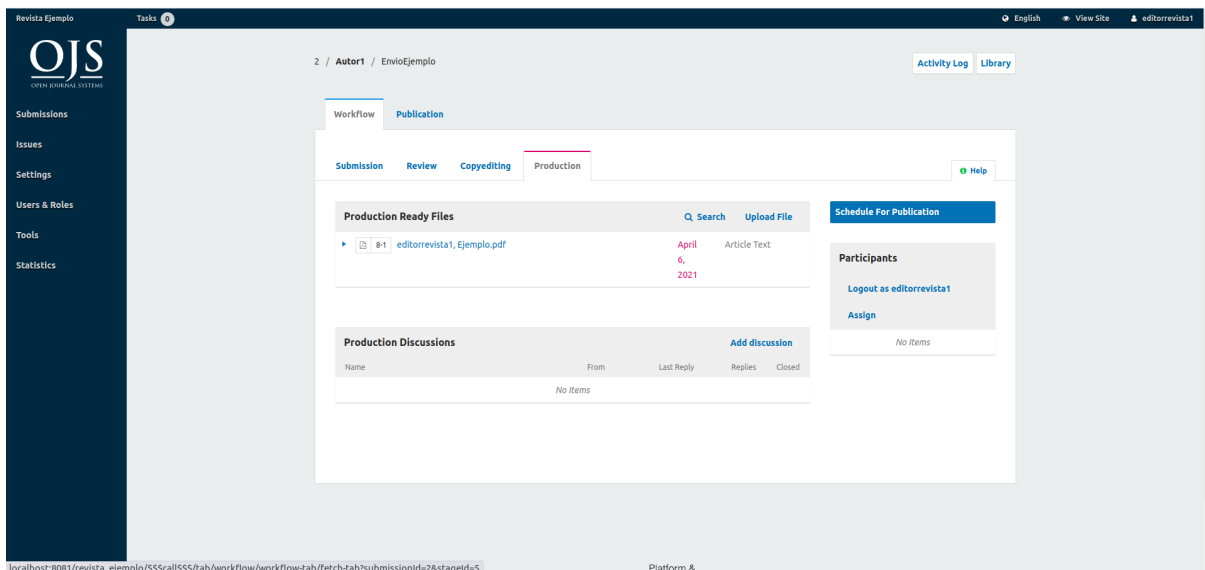
*Imagen 6: Envío en etapa de Envío pendiente (submission) migrado. Etapa de submission luego de ser sometida al proceso de exportación/importación del plugin Native. Fuente: extracción de la aplicación OJS, flujo de trabajo editorial.*



*Imagen 7: Envío en etapa de Revisión (revision) migrado. Etapa de revisión luego de ser sometida al proceso de exportación/importación del plugin Native. Fuente: extracción de la aplicación OJS, flujo de trabajo editorial.*



*Imagen 8: Envío en etapa Editorial (copyediting) migrado. Etapa de copyediting luego de ser sometida al proceso de exportación/importación del plugin Native. Fuente: extracción de la aplicación OJS, flujo de trabajo editorial.*



*Imagen 9: Envío en etapa de Producción (production) migrado. Etapa de production luego de ser sometida al proceso de exportación/importación del plugin Native. Fuente: extracción de la aplicación OJS, flujo de trabajo editorial.*

En las imágenes previas se puede notar que actualmente hay ciertos elementos que no se tienen en cuenta en este proceso de exportación. Estos son:

- Los participantes del proceso editorial de artículo
- Las discusiones
- Todo el proceso de revisión externa, con sus rondas y revisiones

Se debe dejar en claro que el proceso de exportación del plugin *Native* está orientado al artículo en sí. Sin embargo, el desarrollo de este proyecto pretende abarcar todos los puntos anteriormente mencionados y tiene como finalidad que el proceso de exportación englobe no sólo el artículo, sino el envío completo. Los incisos mencionados representan diferentes secciones del envío que son de suma importancia para el proceso editorial. La sección participantes, por un lado, indica quienes están involucrados en el envío y qué rol cumplen en el mismo. La sección de discusiones, por otra parte, es el lugar donde se entablan conversaciones entre los implicados del envío, lo que genera un intercambio de ideas que aportan de manera significativa al desarrollo del artículo durante el proceso editorial. Por último, está la sección de revisión externa, que se caracteriza por ser más compleja. Representa una gran parte del proceso de envío y es donde el artículo se somete a la etapa de revisión por pares. Esta sección está compuesta por rondas y cada ronda está compuesta por una serie de revisiones que se le realizan al artículo.

Conforme este proyecto avance, se explicarán cada uno de los puntos mencionados, cómo están compuestos y las dificultades que conlleva al agregarlos a la exportación.

## Desarrollo de la extensión

Como ya se mencionó en la sección de posibles soluciones, se tomó la decisión de utilizar el plugin *Native* como punto de partida para esta tesis. Dado que este plugin es un proyecto del grupo PKP, marca ciertas pautas de desarrollo a la hora de diseñar/programar una herramienta para OJS. Como se citó anteriormente, esta extensión exporta e importa artículos a través de ficheros XML; estos no son arbitrarios, ya que siguen un esquema determinado por el grupo PKP el que está plasmado en un XSD. Esta sigla corresponde a *XML Schema Definition*, que es un lenguaje utilizado para describir la estructura y las restricciones de los contenidos de los documentos XML de una forma muy precisa, más allá de las normas sintácticas impuestas por el propio lenguaje XML. De esta forma, el XSD determina cómo la sección de exportación debe generar cada XML de los artículos. Este esquema permitió entender cómo PKP ordena y jerarquiza los datos de un artículo para luego plasmarlos en un archivo XML. Aquí se deja un acceso al actual XSD completo del plugin *Native* ([XML native.xsd actual de OJS](#)), sin embargo, para el propósito de esta tesis es necesario identificar específicamente el elemento `articleInfo`, que representa toda la información de un artículo:

```
<complexType name="articleInfo">
  <complexContent>
    <extension base="pkp:submission">

      <attribute name="stage" use="required">
        <simpleType>
          <restriction base="string">
            <enumeration value="submission" />
            <enumeration value="externalReview" />
            <enumeration value="editorial" />
            <enumeration value="production" />
          </restriction>
        </simpleType>
      </attribute>

    </extension>
  </complexContent>
</complexType>
```

*Elemento articleInfo de native.xsd*

En realidad, este elemento es una extensión y una sustitución del elemento submission que ya tenían representado en otro esquema predecesor llamado pkp-native.xsd, indicado por el tag *include* al comienzo del archivo. Es de gran importancia recalcar que esta definición original de submission no representa un envío completo, más allá de su nombre.

```
<complexType name="submission">
  <sequence>
    <element ref="pkp:id" minOccurs="0" maxOccurs="unbounded" />

    <!-- Metadata -->
    <element ref="pkp:submission_file" minOccurs="0" maxOccurs="unbounded" />
    <element ref="pkp:pkppublication" minOccurs="1" maxOccurs="unbounded" />
  </sequence>
  <attribute name="status" type="string" use="optional" />
  <attribute name="current_publication_id" type="int" use="optional" />
  <attribute name="date_submitted" type="date" use="optional" />
```



```
<attribute name="submission_progress" type="int" use="optional" />  
</complexType>
```

#### Elemento *submission* de *pkp-native.xsd*

El elemento *submission* está compuesto por una secuencia de 3 establecidos por PKP, estos son: *id*, *submission\_file* y *pkppublication* y atributos como *status*, *current\_publication\_id*, *date\_submitted*, *submission\_progress*. Al volver hacia la redefinición de este elemento como *articleInfo*, se puede observar que la extensión que se declara en *native.xsd* implica agregar a este elemento un atributo que determina el concepto de etapa. En otras palabras, el elemento *submission* (con toda su estructura ya definida) pasa a llamarse *articleInfo* y se le agrega el atributo *stage* a su estructura que puede tener uno de esos valores que se indican en *native.xsd*. Si bien este esquema es de gran utilidad para generar una exportación a nivel de artículo, aún no es lo suficientemente complejo como para generar una exportación a nivel de envío.

En una primera instancia, fue necesario pensar cómo extender el esquema *native.xsd* de una forma correcta y escalable, para luego generar archivos XML más complejos y que abarquen toda la información adicional que se busque agregar a este proceso de exportación, es decir, pasar de artículo a envío. En este trabajo se creó un archivo adicional XSD que utiliza el original XSD de PKP y se lo extendió para lograr este resultado: [XML native.xsd extendido](#).

Para extender el esquema original, se realizó un proceso iterativo para cada uno de los elementos faltantes, que contó con los siguientes pasos:

- Analizar el elemento faltante y cómo estaba compuesto
- Analizar la forma en que OJS obtiene los datos desde la base de datos
- Utilizar este método para obtener los datos en la etapa de exportación y crear el archivo XML
- Extender el XSD con el objetivo de dar lugar a los nuevos datos
- Extender la funcionalidad de importación para que procese los nuevos datos.

Estos puntos implicaron un análisis de OJS, indagando por un lado su código fuente y cómo estaba constituido su esquema de base de datos, por el otro, cuáles eran sus reglas de negocio respecto a todo el proceso editorial además de entender el funcionamiento del plugin *Native*, su diseño y su integración con OJS.

A partir del nuevo XSD, se explicarán paso a paso todos los nuevos elementos que lo componen. Luego de eso se hará hincapié en problemas que surgen a raíz del agregado de nuevos metadatos al proceso de exportación.

En primer lugar se abordará el elemento *participants* (participantes).

```
<redefine schemaLocation="./native_old.xsd" >
  <complexType name="articleInfo">
    <complexContent>
      <extension base="articleInfo">
        <sequence>
          <element ref="pkp:participants"></element>
          <element ref="pkp:stages"></element>
        </sequence>
      </extension>
    </complexContent>
  </complexType>
</redefine>

<element name="participants">
  <complexType>
    <sequence>
      <element ref="pkp:participant" minOccurs="1" maxOccurs="unbounded" />
    </sequence>
  </complexType>
</element>

<element name="participant">
  <complexType>
    <attribute name="mail" type="string"/>
    <attribute name="user_group_ref" type="string"/>
  </complexType>
</element>
```

En esta sección se pueden observar dos puntos distintos. Por un lado, está la redefinición de *articleInfo* (ya creado en el XSD original de PKP), donde se le agrega, en parte, el elemento *participants*. Más adelante, *participants* se define, de forma tal que está compuesto por una

secuencia de elementos *participant* con una cantidad mínima de 1 y una cantidad máxima indefinida. A su vez, podemos ver como el elemento *participant* posee 2 atributos: *mail* y el *user\_group\_ref*. El *mail* sirve para identificar al usuario a la hora de importarlo y el *user\_group\_ref* indica cual es el rol de ese usuario en el envío de un artículo.

```
<redefine schemaLocation="./native_old.xsd" >
  <complexType name="articleInfo">
    <complexContent>
      <extension base="articleInfo">
        <sequence>
          <element ref="pkp:participants"></element>
          <element ref="pkp:stages"></element>
        </sequence>
      </extension>
    </complexContent>
  </complexType>
</redefine>

<element name="stages">
  <complexType>
    <sequence>
      <element ref="pkp:stage" minOccurs="5" maxOccurs="5" />
    </sequence>
  </complexType>
</element>

<element name="stage">
  <complexType>
    <sequence>
      <element ref="pkp:queries" />
      <element ref="pkp:rounds" minOccurs="0" maxOccurs="1" />
    </sequence>
    <attribute name="id" />
    <attribute name="name" type="string" />
  </complexType>
```

```
/element>
```

Por otra parte, se puede ver que, en la misma secuencia donde se agregaron los participantes, también se incorporó el elemento *stages*, conformado por elementos *stage*, y cada *stage* posee una secuencia con elementos *queries* y *rounds*, además de atributos como el *id* del *stage* y su nombre. Esto representa cada etapa del proceso de envío del artículo (*submission*, *external review*, *copyediting*, *production*).

```
<element name="queries">
  <complexType>
    <sequence>
      <element ref="pkp:query" minOccurs="0" maxOccurs="unbounded" />
    </sequence>
  </complexType>
</element>

<element name="query">
  <complexType>
    <sequence>
      <element ref="pkp:note" minOccurs="1" maxOccurs="unbounded" />
    </sequence>
  </complexType>
</element>

<element name="note">
  <complexType mixed="true">
    <attribute name="user" type="string"/>
    <attribute name="date_created"/>
    <attribute name="date_modified"/>
    <attribute name="title"/>
  </complexType>
</element>
```

El elemento *queries* está compuesto por una secuencia de elementos *query*, y es opcional. Cada *query*, posee 1 o más atributos *note*. Este último tiene atributos como el usuario (quien lo hizo),

fecha de creación, fecha de modificación, un título y, al ser un elemento de tipo mixto, el cuerpo de la nota está dentro del mismo.

Cada *query* representa una *discussion* y cada *note* dentro de esa *query* es un mensaje en la *discussion*.

```
<element name="rounds">
  complexType>
  <sequence>
    <element ref="pkp:round" minOccurs="0" maxOccurs="unbounded" />
  </sequence>
</complexType>
</element>

<element name="round">
  complexType>
  <sequence>
    <element ref="pkp:reviewAssignment" minOccurs="0" maxOccurs="unbounded" />
    <element ref="pkp:file" minOccurs="0" maxOccurs="unbounded" />
  </sequence>
</complexType>
element>
```

Como se mencionó, el elemento *stage* está conformado por *rounds*, que es opcional (esto es importante ya que la etapa de revisión externa es la única que tiene rondas). Este elemento, a su vez, está compuesto por *round*. Este último posee 2 grandes elementos: *file* y *reviewAssignment*, ambos opcionales.

```
<element name="file">
  complexType>
  <sequence>
    <element ref="pkp:name"/>
    <element ref="pkp:embed" minOccurs="0"/>
  </sequence>
  <attribute name="id" />
  <attribute name="number" />
```

```
<attribute name="stage" />
<attribute name="source" />
<attribute name="genre" />
<attribute name="filename" />
<attribute name="viewable" />
<attribute name="date_uploaded" />
<attribute name="date_modified" />
<attribute name="filesize" />
<attribute name="filetype" />
<attribute name="uploader" />
/complexType>
element>

<element name="name">
complexType mixed="true">
<attribute name="locale" />
/complexType>
element>

element name="embed">
complexType mixed="true">
<attribute name="encoding" />
/complexType>
element>
```

Por un lado, está el elemento *file* el cual representa al archivo que en la ronda de revisión, posee varios atributos como *id*, *number*, *filesize*, *filetype*, etc. Además, de 2 elementos: *name* y *embed*. *Name* representa el nombre del archivo con un atributo *locale* para indicar el idioma y *embed* contiene el archivo codificado y el atributo *encoding* indica cual es la codificación que se utilizó.

```
<element name="reviewAssignment">
complexType>
<sequence>
<element ref="pkp:form" minOccurs="1" />
</sequence>
```

```
<attribute name="reviewer" />
<attribute name="method" />
<attribute name="round" />
<attribute name="unconsidered" />
<attribute name="date_rated" />
<attribute name="last_modified" />
<attribute name="date_assigned" />
<attribute name="date_notified" />
<attribute name="date_confirmed" />
<attribute name="date_completed" />
<attribute name="date_acknowledged" />
<attribute name="date_reminded" />
<attribute name="date_due" />
<attribute name="date_response_due" />
<attribute name="declined" />
<attribute name="cancelled" />
<attribute name="automatic" />
<attribute name="quality" />
<attribute name="recommendation" />
<attribute name="competing_interest" />

/complexType>
element>

- Form -->

<element name="form">
  complexType>
  <sequence>
    <element ref="pkp:answer" minOccurs="0" maxOccurs="unbounded" />
  </sequence>
  <attribute name="title" />
/complexType>
element>

element name="answer">
```

```
complexType>  
<attribute name="value" />  
<attribute name="viewable" />  
/complexType>  
element>
```

Por otro lado, el elemento *reviewAssignment* representa las revisiones que se le asignaron a diferentes usuarios para el envío. Posee atributos como *reviewer* que indica quien es el revisor, *method* para indicar el tipo de revisión (abierta, doble ciega, ciega), distintos atributos de fechas para ver en qué estado está la revisión (solicitada, resuelta, etc.), *quality* que indica la puntuación de la revisión, entre otros. Además, *reviewAssignment* está compuesto por el elemento *form* el que representa el formulario que se utilizó en la revisión. OJS permite crear formularios predeterminados para que los revisores sigan una línea de trabajo puntual. En caso de que no se utilice ningún formulario creado, OJS provee uno por defecto.

*Form* posee un atributo *title* que indica el nombre del formulario para identificarlo. Además, lo compone el elemento *answer*, que representa las respuestas que fueron cargadas por los revisores para cada pregunta del formulario. Poseen los atributos *value*, como respuesta, y *viewable*, que indica quién (sólo el editor, o el editor y los autores) puede ver la respuesta (dependiendo formulario).

Para obtener todos estos metadatos e incorporarlos al proceso de exportación, fue necesario entender cómo se obtienen en el uso cotidiano de OJS. Se observó que OJS utiliza el patrón DAO para acceder a la información de los objetos, y para asignarla. El patrón *Data Access Object* proporciona una interfaz abstracta a algún tipo de base de datos u otro mecanismo de persistencia. Al asignar las llamadas de la aplicación a la capa de persistencia, el DAO proporciona algunas operaciones de datos específicas sin exponer los detalles de la base de datos. De cada elemento que se agregó al proceso de exportación, toda su información fue extraída a través de su DAO correspondiente. Utilizar este patrón tiene sus ventajas: simplifica la extracción y seteo de datos, ya que existen métodos predefinidos que hacen este trabajo, y el hecho de que se utilicen asegura que sea de forma correcta, pues estos DAO son desarrollados por el grupo PKP y se respetan las reglas de negocio planteadas por ellos. Además, en caso de que los DAO cambiaran internamente sus reglas de negocio no afectaría a esta extensión (con excepción de que los métodos cambien drásticamente).



Una vez analizada la ventaja de los DAO, por un lado, estos se utilizaron en el proceso de exportación para acceder a los datos. Su uso resuelve parte del proceso iterativo para extender el plugin que se mencionó anteriormente. Por otro lado, una vez que se hallaba el DAO correspondiente al punto faltante que se quería agregar, se utilizaba para obtener la información necesaria para luego plasmarla en un archivo XML.

Estos mismos DAO fueron empleados también en el proceso de importación, que consistía en recorrer los nuevos nodos del XML y utilizar cada DAO correspondiente al elemento para crear las nuevas entidades en otra instalación de OJS. Para este paso, fue necesario estudiar cómo es el proceso de creación de cada elemento en el uso cotidiano de OJS y cómo interactúan los DAO.

Es importante aclarar que en el momento que se tomó la decisión de extender este plugin, se decidió utilizarlo en la última versión estable de OJS (versión 3.2.1-1). Esto constituye un punto clave, ya que el esquema definido de esta versión y la forma de acceder a los metadatos cambia con respecto a versiones anteriores. Esto sugiere que, en caso de que se quiera hacer uso de esta nueva extensión del plugin, es necesario que la instalación de OJS, origen y destino pertenezcan a esta versión. Si bien esto es una limitación, más adelante en el trabajo en la sección de Desarrollos a futuro, se propondrán alternativas que pueden sortear este tipo de problemáticas.

Para concluir este capítulo, se expuso la forma como quedaría extendido el plugin para la exportación/importación de envíos. Se hizo hincapié en cómo se obtuvo la información, cómo se procesó y de qué forma queda extendido XSD para la creación de archivos XML. Sin embargo, con toda esta cantidad de información adicional surgen una serie de conflictos que hay que tener en cuenta y resolver.

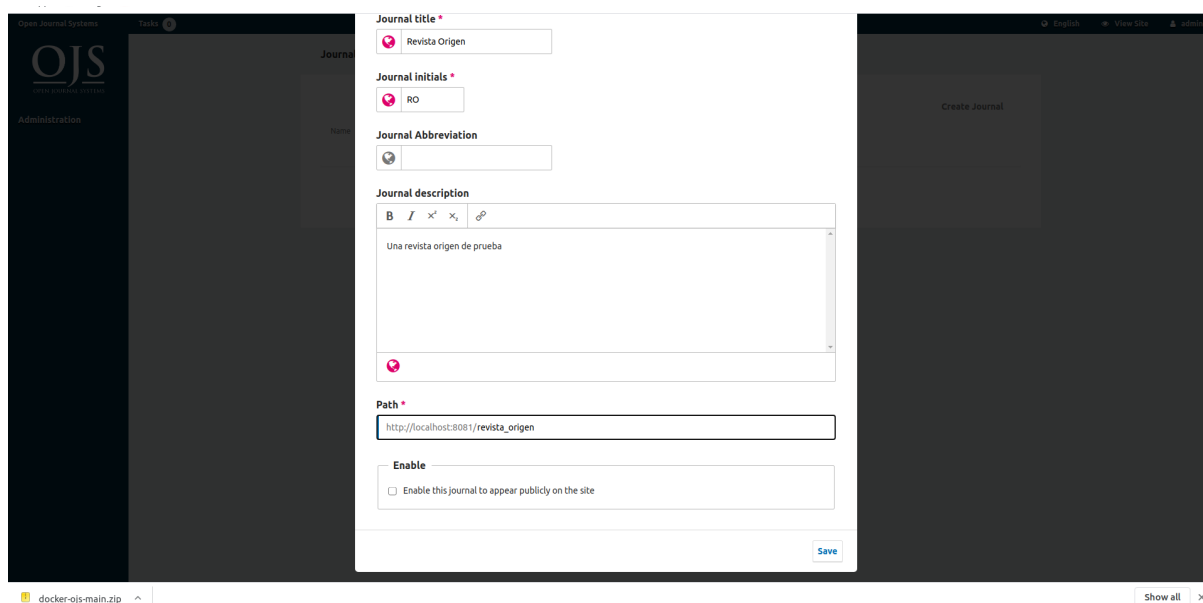
Como se puede notar, en muchos de los elementos están involucrados usuarios del sistema (*participants, notes, reviewAssignments*). A su vez, también existen diferentes roles para los usuarios y diferentes formularios personalizados. Todos estos incisos tienen algo en común: en caso de exportar a una nueva instalación de OJS, todos estos pueden estar ausentes. Por ejemplo, como se mencionó anteriormente, los usuarios están identificados con una dirección de correo electrónico, sin embargo, si la nueva instalación de OJS no tiene el usuario con ese email determinado, la exportación no puede realizarse correctamente. Lo mismo sucede con los roles definidos y los formularios personalizados. En general, para solucionar estas problemáticas

es necesario cargar en el OJS destino todos los usuarios, roles y formularios personalizados faltantes que posee el OJS origen. Para el caso de los usuarios existe un plugin nativo de PKP que proporciona una exportación automática de los mismos, lo que solucionaría la falta de usuarios en el OJS destino, siempre y cuando el plugin respete el correo del usuario (lo que lo identifica en dicha exportación). En el caso de los roles de los usuarios, es necesario que sean replicados manualmente en el OJS destino; lo mismo sucede para los formularios personalizados, estos deben estar replicados con el mismo nombre para que la importación sea exitosa.

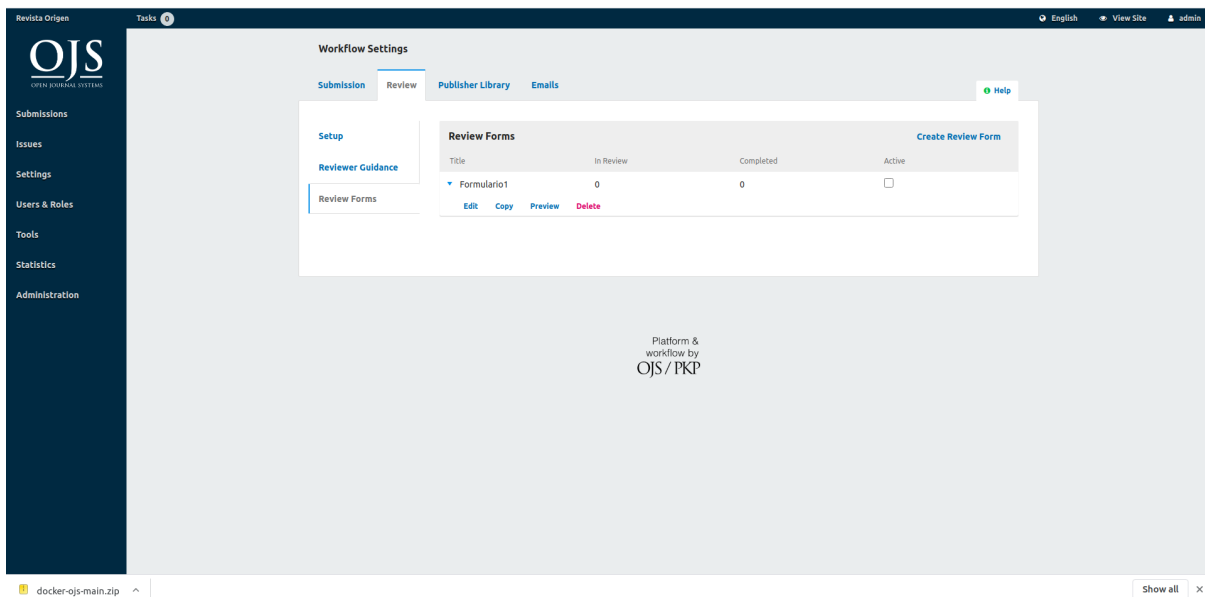
Para terminar, quedó presentado todo el proceso de exportación de envíos de la nueva extensión del plugin *Native*, su análisis, sus alternativas, su desarrollo, sus problemáticas y sus soluciones. En el siguiente capítulo se mostrará cómo fue diseñada la prueba de campo, detalladamente, y se hará foco en el proceso de exportación del plugin extendido, junto a un protocolo que se ejecutará previamente para solucionar estas problemáticas y poder realizar una exportación del envío exitosa.

## Capítulo 4 - Pruebas de migraciones

En este capítulo se expondrán distintos casos de uso en donde se utilizó el plugin extendido, los resultados obtenidos y, las condiciones y pasos que se requieren para obtener dichos resultados. Para realizar las pruebas, se creó un entorno controlado, contemplando diferentes casos, para comprobar su correcta funcionalidad. La creación del entorno implicó generar una instalación de OJS en la versión correspondiente desde cero, a la que se le asignó una nueva revista (revista origen), se crearon nuevos usuarios con roles determinados y formularios personalizados (estos ítems resultan importantes, más adelante, a la hora de crear la instalación OJS destino).



*Imagen 10: Configuración base de la revista origen, el cual contiene título de la misma, sus iniciales, la ruta para determinar cómo acceder a la revista y algunos campos opcionales como las abreviaturas y la descripción. Fuente: extracción de la aplicación OJS, configuración de revista.*



*Imagen 11: Sección para crear formularios personalizados. Fuente: extracción de la aplicación OJS, configuración de formularios.*

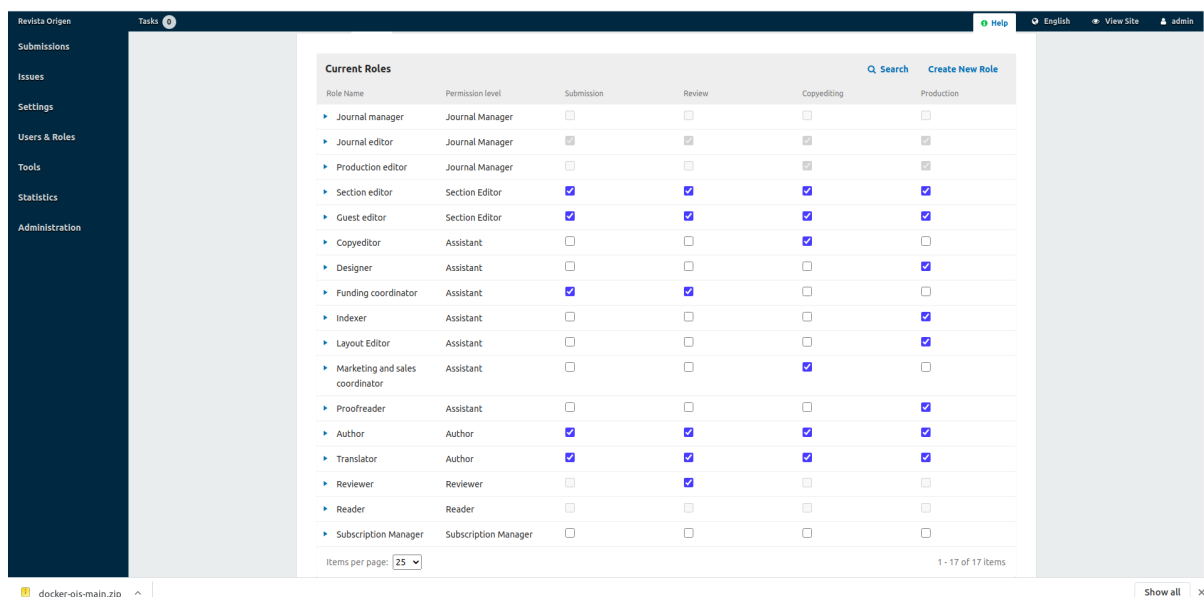
Aquí OJS provee un espacio para que el usuario pueda crear, editar, activar/desactivar y borrar formularios personalizados. Para ello, OJS le provee al usuario la posibilidad de crear preguntas/incisos en los cuales él mismo puede categorizar la respuesta que le corresponde; las opciones de estas categorías varían, pero entre ellas se encuentran: respuesta de una palabra, texto completo, checkboxes, *radio option*, etc. Asimismo, el usuario puede indicar si estas respuestas son opcionales u obligatorias. Por ejemplo, un usuario puede generar un formulario personalizado con la siguiente estructura:

- Pregunta/Inciso 1: Aprobado?
  - Categoría: Single word answer, Obligatoria: Si
- Pregunta/Inciso 2: Justificar
  - Categoría: Textarea, Obligatoria: Si
- Pregunta/Inciso 3: Recomendaciones
  - Categoría: Checkboxes, Obligatoria: No
    - Opción 1: Mejorar caligrafía
    - Opción 2: Ajustar marginado

Un modelo de formulario completado podría ser de la siguiente manera:

- Aprobado?
  - “Sí”
- Justificar
  - “La capacidad que tiene para expresar sus ideas excede los límites de la comprensión humana”
- Recomendaciones
  - Ajustar marginado

Como se mencionó en el capítulo anterior, todas estas posibles estructuras para formularios están contempladas en la nueva extensión.



Role Name	Permission level	Submission	Review	Copyediting	Production
Journal manager	Journal Manager	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Journal editor	Journal Manager	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Production editor	Journal Manager	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Section editor	Section Editor	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Guest editor	Section Editor	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Copyeditor	Assistant	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Designer	Assistant	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Funding coordinator	Assistant	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Indexer	Assistant	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Layout Editor	Assistant	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Marketing and sales coordinator	Assistant	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Proofreader	Assistant	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Author	Author	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Translator	Author	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Reviewer	Reviewer	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Reader	Reader	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Subscription Manager	Subscription Manager	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

Imagen 12: Sección de roles. Fuente: extracción de la aplicación OJS, configuración de roles.

Cómo se observa en la imagen 12, OJS posee una sección de configuración donde se indican todos los roles (y permisos de los mismos) que se quieren designar en la revista. Hay que recordar que no todas las instituciones o equipos editoriales trabajan de igual manera, por lo que esta capacidad que provee OJS es de gran utilidad.

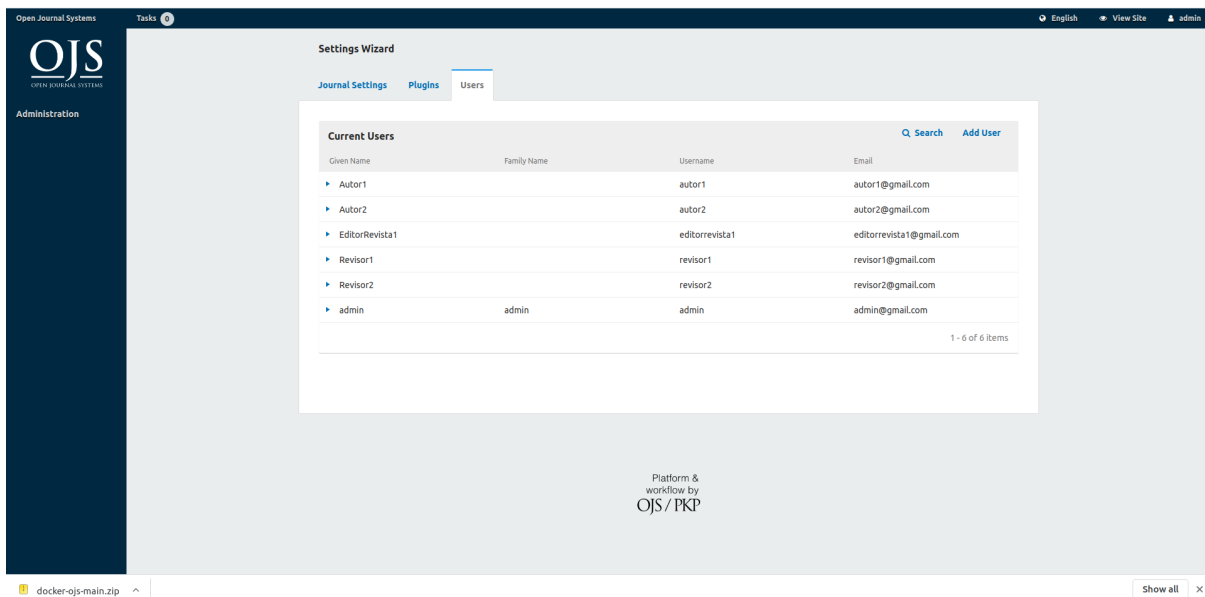


Imagen 13: Sección de usuarios del sistema. Fuente: extracción de la aplicación OJS, configuración de usuarios.

La figura 13 muestra la sección donde se guardan todos los usuarios que intervienen en la revista y allí se indica a qué rol están asociados (autor, revisor, editor, etc).

## Prueba de campo

Una vez preparado el entorno, se crearon varios tipos de envíos en distintas etapas, con diferentes usuarios que intervinieron en cada proceso, y con diferentes archivos. Todo esto, con el fin de generar una exportación y luego integrar en la instalación destino de OJS. Después de haber generado los envíos, se integró el nuevo plugin a la instalación de OJS y se ejecutó para generar una exportación de cada uno de ellos. Al ser una extensión del plugin *Native*, la forma de utilizarlo es la misma que la del original. Este se encuentra en el menú de herramientas, en la sección import/export. Allí se encuentran los diferentes plugins asociados a exportaciones y, entre ellos, está el nuevo plugin *Native*. Una vez seleccionado facilita la opción de exportar o importar un envío.

A continuación, se mostrarán diferentes casos de exportación e importación de envíos, en diferentes etapas del proceso editorial, con diferentes metadatos. Cabe recalcar que el proceso de exportación tiene en cuenta todas las etapas de cada envío, por lo que, todo lo que se muestre en una etapa de un envío, se mantendrá en caso de que este pase a otra etapa.

## Caso 1: Envío en etapa de *submission*

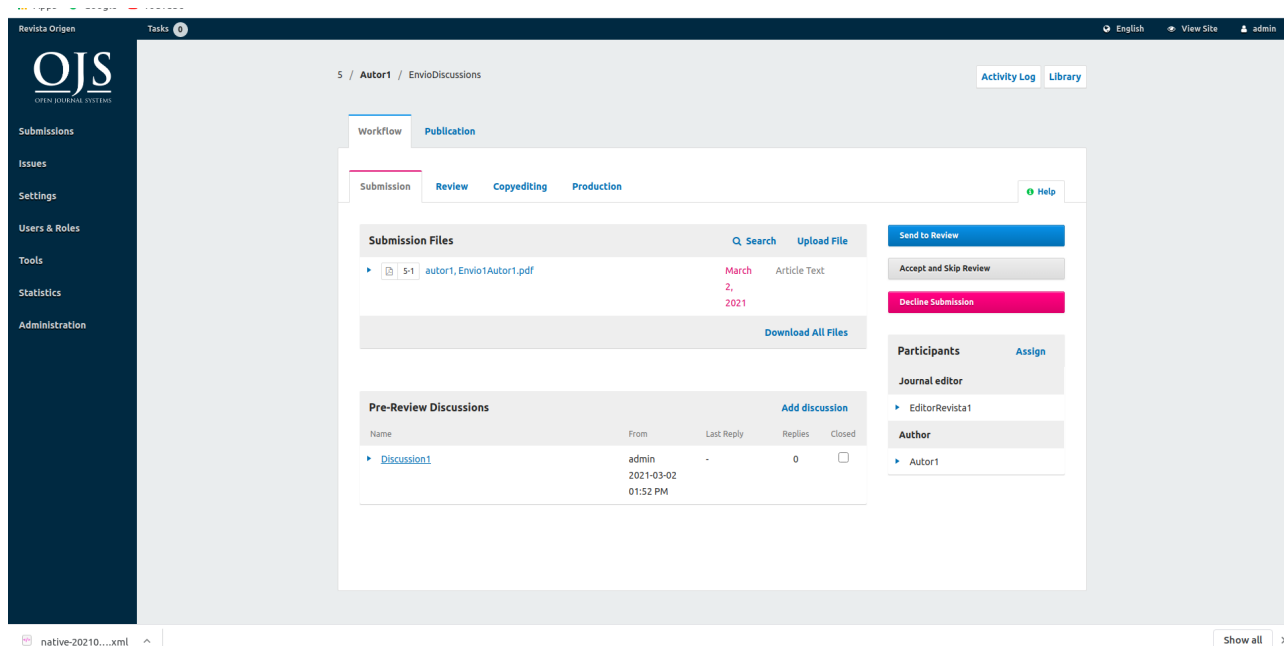
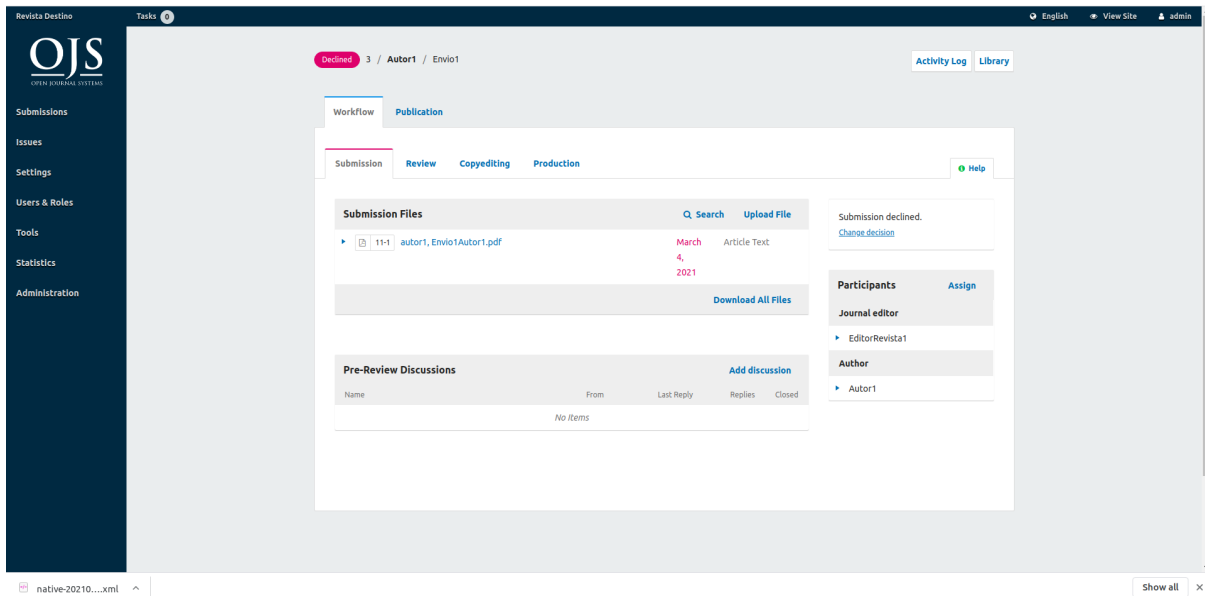


Imagen 14: Envío en etapa de Envío pendiente (*submission*) migrado. Etapa de *submission* luego de ser sometida al proceso de exportación/importación del plugin Native extendido. Fuente: extracción de la aplicación OJS, flujo de trabajo editorial.

Este caso es el más simple: un envío que entró en etapa de *submission*. Como se puede apreciar, el proceso de exportación tiene en cuenta los *submission files*, los participantes del envío y las discusiones de esta etapa. Por un lado, los *submission files* ya estaban contemplados originalmente en el plugin de *Native* y, por otro, remontándose al capítulo 3, donde se habló de *XSD* extendido y su estructura, se puede observar que intervienen diversos elementos del XML, como por ejemplo `<participants>`, que engloba la estructura de los participantes, y `<queries>`, que abarca la información de las discusiones. No obstante, el elemento `<stage>` juega un rol importante (para este caso y en los siguientes), ya que define qué metadatos corresponde a cada etapa del proceso editorial. En este caso, las discusiones que se muestran en la imagen pertenecen solamente a la etapa de *submission*.

## Caso 2: Envío cancelado



*Imagen 15: Envío cancelado migrado. Envío cancelado luego de ser sometido al proceso de exportación/importación del plugin Native extendido. Fuente: extracción de la aplicación OJS, flujo de trabajo editorial.*

Similar al caso anterior, aquí un envío pasó por la etapa de submission y fue rechazado por algún motivo. Tiene la particularidad de que no fue necesario agregar algún tipo de funcionalidad al desarrollo actual del plugin *Native* para contemplarlo. Como se mencionó antes, al aprovechar la extensibilidad que provee el formato XSD se pudieron agregar metadatos adicionales (como participantes, discusiones, etc.) al proceso de exportación, conservando las funcionalidades que ya proveía el plugin, y mantener el estado de los artículos es una de ellas. Esta es la primera prueba de avance respecto a cualquier otra de las alternativas que se mencionaron en el capítulo 3, donde se muestra la posibilidad de migrar envíos de artículos rechazados.



### Caso 3: Envío en *revision*

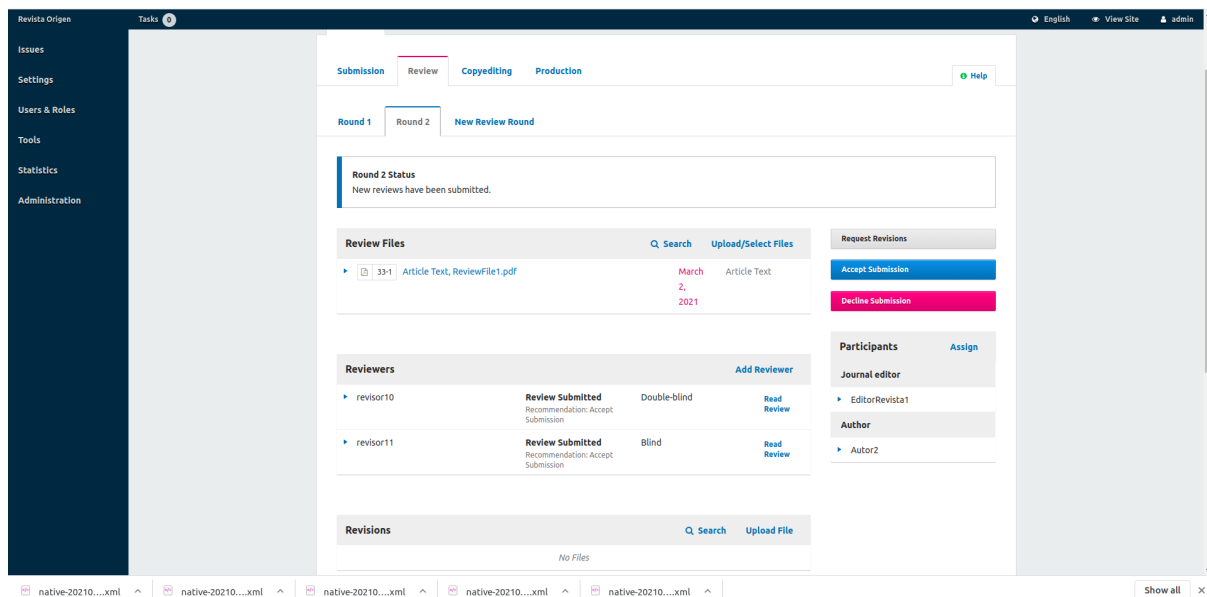
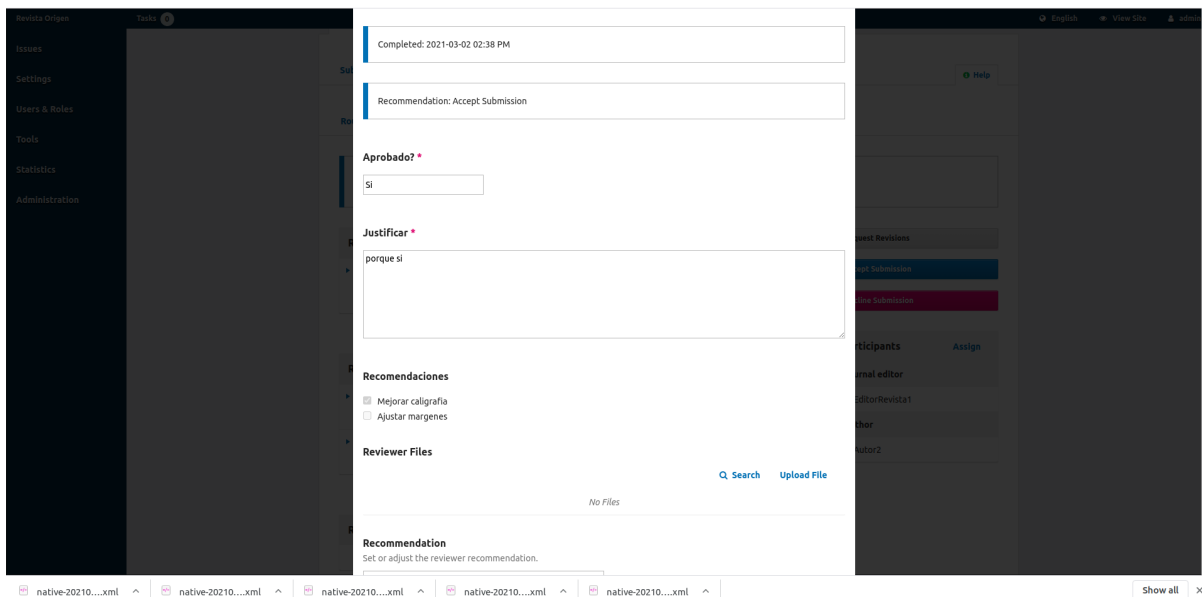


Imagen 16: Envío en etapa de Revisión (*revision*) migrado. Etapa de *revision* luego de ser sometida al proceso de exportación/importación del plugin Native extendido. Fuente: extracción de la aplicación OJS, flujo de trabajo editorial.

En este caso, los metadatos del envío se tornan más complejos. Luego de haber sido aceptado, el envío pasa a la etapa de revisión, conformada por rondas, revisores, revisiones y discusiones (y como se aclaró anteriormente, incluye toda la información de la etapa anterior). Es imperativo remarcar que, antes de este proyecto, toda esta etapa no estaba incluida en el proceso de exportación.

Como ya se mencionó en los capítulos anteriores, el proceso editorial está conformado, en parte, por una etapa de revisión por pares y en este caso se representa la misma. Se puede notar que la etapa de revisión (en la que interviene el elemento `<stage>` nuevamente) está conformada por rondas representadas por el elemento `<round>`, mencionado en el capítulo 3. Se espera que en cada una de estas rondas de revisión sean asignados revisores (se puede ver en la imagen que para la ronda 2 hay 2 revisores asignados) y estas son representadas por el elemento `<reviewAssignment>`. Cada una de estas revisiones puede tener estados distintos, dependiendo del trabajo del revisor (revisión asignada, aceptada, realizada, etc). Todos estos casos están contemplados en el proceso de exportación y registrados por medio de fechas, que tienen asignados los `<reviewAssignment>` como metadatos (`last_modified`, `date_assigned`, `date_notified`, `date_completed`, etc). Con esta información, la misma aplicación de OJS evalúa en

qué estado se encuentra el <reviewAssignment> e informa el resultado mediante un mensaje en pantalla.



*Imagen 17: Formulario personalizado migrado. Uno de los tantos formularios que pueden ser utilizados en revisiones luego de ser sometido al proceso de exportación/importación del plugin Native extendido.*

*Fuente: extracción de la aplicación OJS, flujo de trabajo editorial.*

Además, se puede observar que en la etapa de revisión se hace uso de formularios de revisión personalizados, representados por <form>, que también fueron agregados a la exportación (y en caso de que no se utilicen, se hace uso del formulario predefinido por OJS). Como se mencionó al principio de este capítulo, es imperativo que los formularios que se quieran integrar al proceso estén replicados de manera idéntica en el OJS destino, respetando tanto su nombre como los campos que componen cada formulario (con el mismo nombre, mismo tipo, mismas opciones de configuración y mismo orden en la definición).

## Caso 4: Envío en *copyediting*

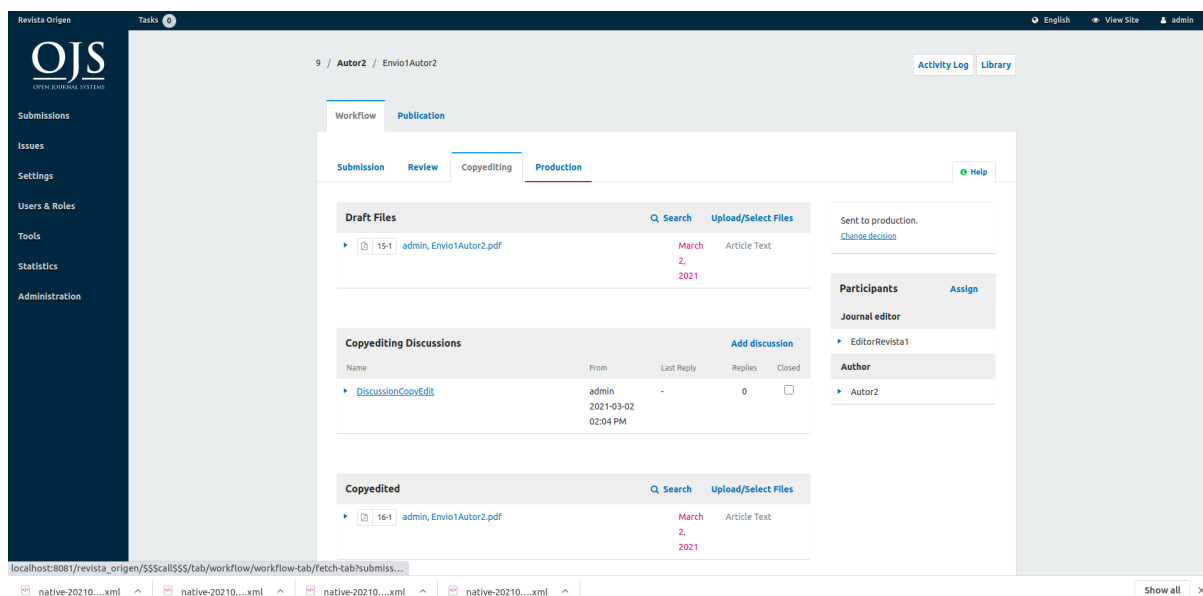
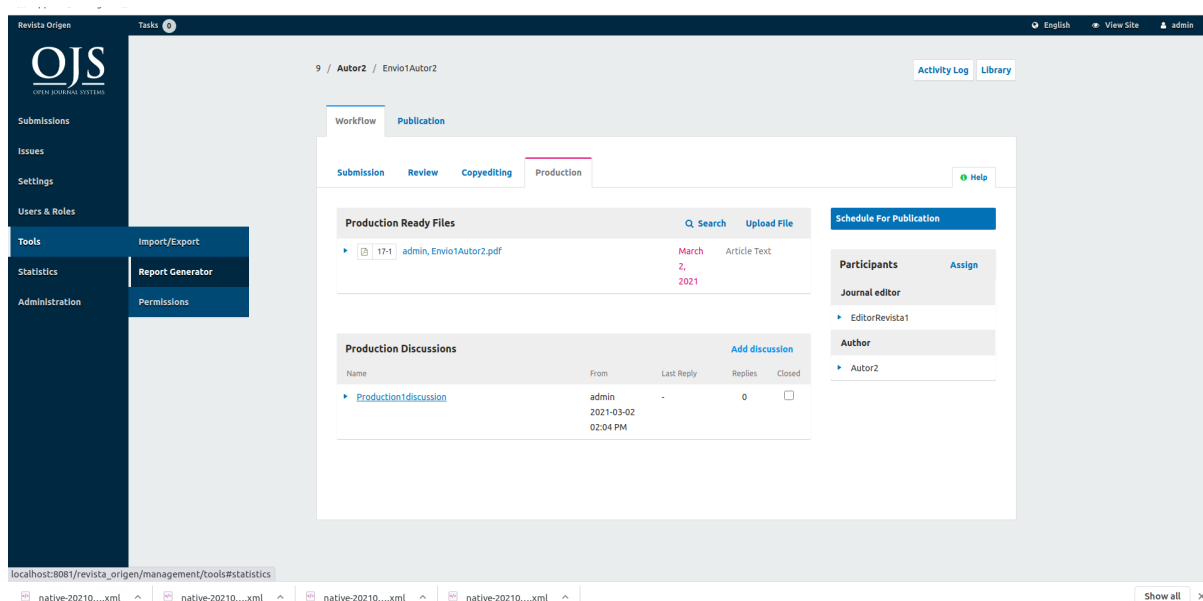


Imagen 18: Envío en etapa Editorial (*copyediting*) migrado. Etapa de *copyediting* luego de ser sometida al proceso de exportación/importación del plugin Native extendido. Fuente: extracción de la aplicación OJS, flujo de trabajo editorial.

Una vez que un envío pasa la etapa de revisión, es dirigido a la etapa de *copyediting*. Esta es más simple que su predecesora y se asimila a la etapa de *submission*. Como todas, por un lado, tiene los archivos resultantes de la etapa de revisión, que son los que serán usados en esta etapa para hacer el proceso de *copyediting*. Estos quedan guardados como *copyediting* files, que están incluidos en la migración. Conjuntamente, esta etapa también posee discusiones, en donde el autor y los usuarios pueden intercambiar ideas correspondientes a la etapa.

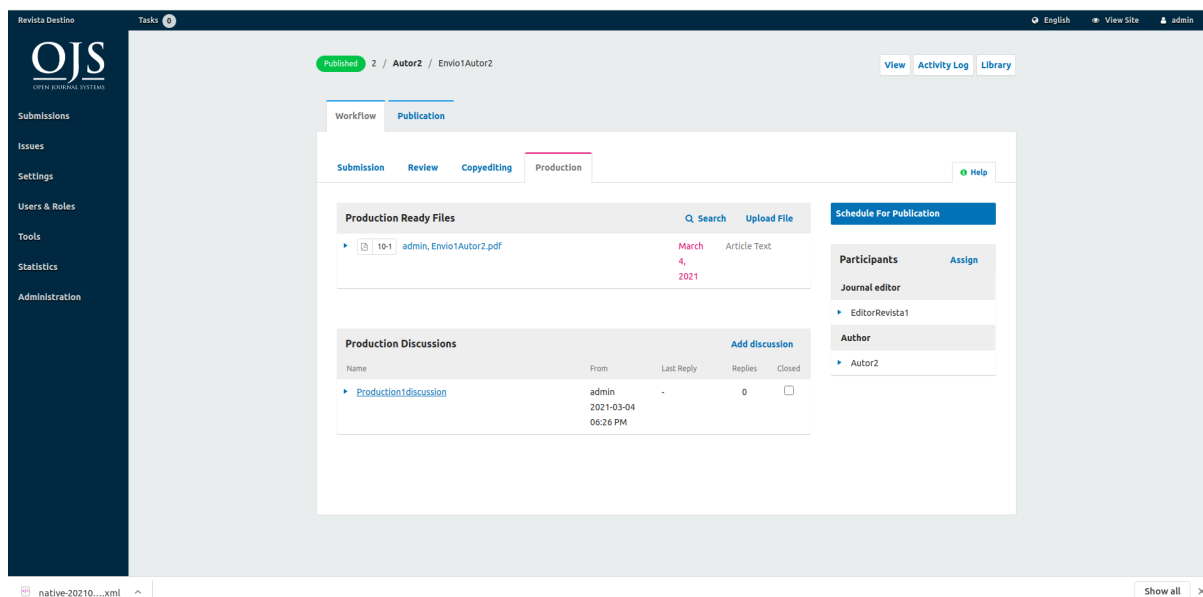
## Caso 5: Envío en *production*



*Imagen 19: Envío en etapa de Producción (production) migrado. Etapa de production luego de ser sometida al proceso de exportación/importación del plugin Native extendido. Fuente: extracción de la aplicación OJS, flujo de trabajo editorial.*

Similar a la etapa de *submission* y *copyediting*, *production* tiene archivos de entrada provenientes de *copyediting*, los archivos generados para la publicación y las discusiones.

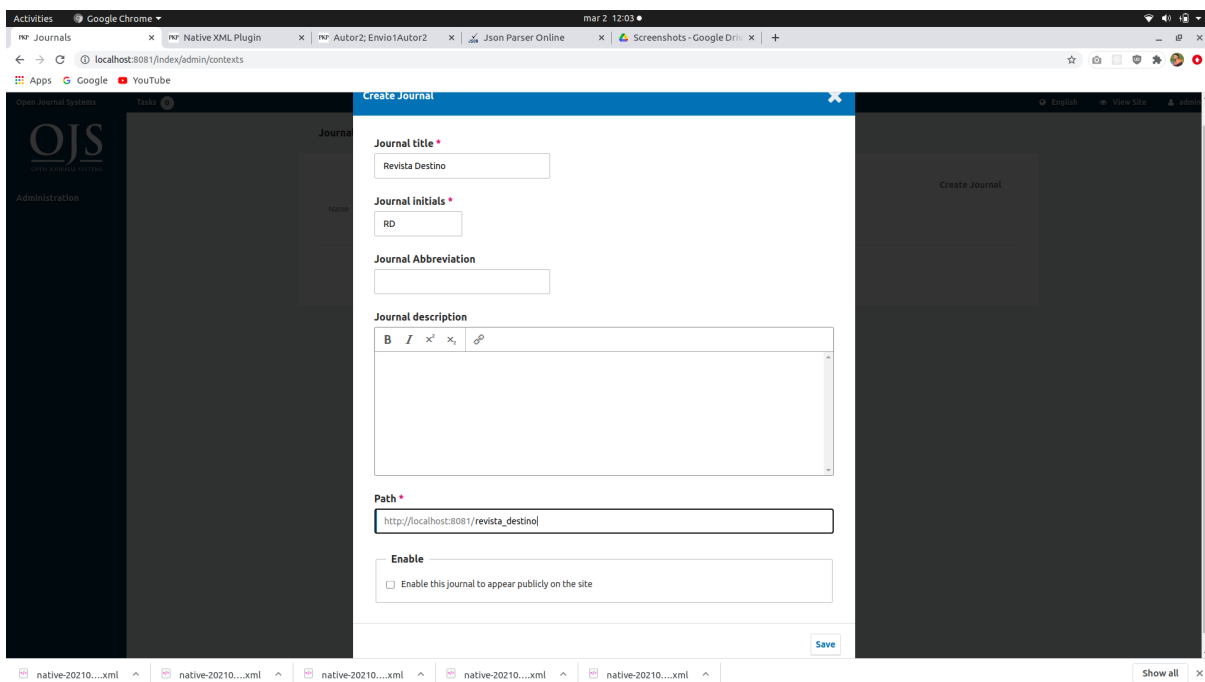
## Caso 6: Envío publicado



*Imagen 20: Envío publicado migrado. Envío publicado luego de ser sometido al proceso de exportación/importación del plugin Native extendido. Fuente: extracción de la aplicación OJS, flujo de trabajo editorial.*

Este es el caso de un envío que llegó a la etapa de *production* y luego fue publicado. Al igual que el caso 2, el estado de “publicado” de un artículo ya estaba contemplado en el plugin *Native*.

Con todos los envíos creados, se utilizó el nuevo plugin con el fin de realizar el proceso de exportación para generar un archivo xml por cada envío. Una vez almacenados cada uno de estos, se creó una nueva instalación de OJS respetando, como se mencionó anteriormente, la configuración del OJS origen.



*Imagen 21: Configuración base de la revista destino, el cual contiene título de la misma, sus iniciales, el path para determinar cómo acceder a la revista y algunos campos opcionales como las abreviaturas y la descripción. Fuente: extracción de la aplicación OJS, configuración de revista.*

Una vez creada la nueva instalación, se utilizó la funcionalidad de importación del nuevo plugin con el objetivo de importar cada envío exitosamente. Como se mencionó anteriormente, la forma de uso del plugin es exactamente igual que el original.

Se destaca que los tiempos de espera tanto al exportar como al importar son despreciables, ya que se logró que tanto la exportación como la importación demoren poco tiempo, apenas unos

segundos; y es necesario tener en cuenta que los envíos fueron exportados e importados uno en uno (generando un XML por cada envío). A su vez, si se aprovecha la escalabilidad del XSD al extender el plugin, se puede utilizar la funcionalidad de exportar todos los envíos en un solo XML. Esto implica que el usuario puede importar este archivo XML, compuesto por muchos envíos en una sola acción lo que acelera el proceso de exportación/importación significativamente. A continuación, se muestra un ejemplo de un XML resultado, compuesto por muchos envíos:

```
<?xml version="1.0"?>
<articles xmlns="http://pkp.sfu.ca"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://pkp.sfu.ca native.xsd">
  <article xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
date_submitted="2021-03-02" status="3" submission_progress="0"
current_publication_id="1" stage="production">...
  </article>
  <article xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
date_submitted="2021-03-01" status="4" submission_progress="0"
current_publication_id="3" stage="submission">...
  </article>
  <article xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
date_submitted="2021-03-02" status="3" submission_progress="0"
current_publication_id="2" stage="production">
  <id type="internal" advice="ignore">2</id>
  <submission_file xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
stage="submission" id="6" xsi:schemaLocation="http://pkp.sfu.ca native.xsd">
    ...
  </submission_file>
  <submission_file xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
stage="review_file" id="7" xsi:schemaLocation="http://pkp.sfu.ca native.xsd">
    ...
  </submission_file>
  <submission_file xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
stage="final" id="8" xsi:schemaLocation="http://pkp.sfu.ca native.xsd">
    ...
  </submission_file>
  <submission_file xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
stage="copyedit" id="9" xsi:schemaLocation="http://pkp.sfu.ca native.xsd">
    ...
  </submission_file>
  <submission_file xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
stage="production_ready" id="10" xsi:schemaLocation="http://pkp.sfu.ca
native.xsd">
    ...
  </submission_file>
  <publication xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
```

```
locale="en_US" version="1" status="3" primary_contact_id="2" url_path="" seq="0"  
date_published="2021-03-04" section_ref="ART" access_status="0"  
xsi:schemaLocation="http://pkp.sfu.ca native.xsd">  
  ...  
  </publication>  
  <participants>  
    ...  
  </participants>  
  <stages>  
    ...  
  </stages>  
</article>  
</articles>
```

*XML con muchos artículos simplificado*

En este ejemplo simplificado se puede observar que cada `<article>` representa un envío y cada uno de estos *tags* compone al *tag* `<articles>`, siendo este el que representa la colección de estos envíos (en la sección [XML compuesto por muchos artículos](#) se adjunta el XML completo). Al importar, el plugin navega cada *tag* y lo importa de la misma forma que lo haría con un envío único.

Para finalizar, en este capítulo se mostró cómo hacer uso del nuevo plugin. A través de pruebas en un ambiente controlado, se visualizaron sus precondiciones de uso, sus mejoras y los tiempos de ejecución. En el capítulo siguiente, se hará un repaso y conclusión del proyecto, se mencionarán ventajas del mismo, aunque también, se hará foco en mejoras que pueden aplicarse a esta nueva extensión y se propondrán desarrollos a futuro que surgen a raíz de este desarrollo.

## Capítulo 5 - Conclusiones y Trabajos Futuros

A lo largo de esta tesina se expuso una problemática que afecta a diferentes portales de revistas soportados por OJS donde, por diversos motivos, uno desea migrar diferentes artículos, o incluso revistas completas, hacia otro portal o instalación de OJS. Se presentó la alternativa con que la plataforma cuenta actualmente para solucionar la problemática antes mencionada (*Native import/export plugin*). Sin embargo, quedó en evidencia que este plugin estaba focalizado solo en el artículo *per se*, y no así, en todo su proceso editorial. Si bien este módulo ha aportado una solución a gran parte a los diversos usuarios de OJS y sus portales, el hecho de que el proceso editorial no se contemple en el proceso de exportación de un artículo resulta en soluciones incompletas para las entidades que deseen realizar una migración integral. Esta migración incompleta implica que una misma entidad mantenga más de un OJS y no tenga la posibilidad de integrarlos entre sí, sin perder una vasta cantidad de datos. Esta situación puede darse cuando una institución ha creado, a lo largo del tiempo, diferentes instalaciones OJS para manejar sus revistas, sin embargo, luego se percata de que en su caso probablemente sea más acorde unificar todas las revistas en una sola instalación (ya sea por costos de mantenimiento o de administración). También, se puede dar el caso de que una revista pueda pasar de ser administrada por una entidad a otra, lo que implicaría que, si esta otra entidad tiene su propio OJS, se debería llevar a cabo una migración de la revista. Por ejemplo, suponiendo que la UNLP maneja una instalación OJS para todas las revistas de la Facultad de Ingeniería y otro para las revistas de la Facultad de Informática, se puede dar el caso en el que se resuelva que una revista que pertenece Ingeniería deba ser mantenida por Informática (o viceversa). Por situaciones como estas, luego de un largo análisis de diferentes aproximaciones/alternativas para su realización, este proyecto propuso una solución que concluyó en extender este plugin para que abarque el proceso editorial completo del artículo, lo que implicaría una exportación del artículo desde su envío hasta su publicación.

Una vez tomada la decisión de extender el plugin, fue necesario analizar tanto su funcionamiento como su estructura para entender cómo este lograba realizar el proceso de exportación. Teniendo en consideración que el plugin se basaba en la creación de archivos XML delimitados por un XSD, fue imperativo que este esquema se estudiara para entender cómo definió y jerarquizó PKP todos los elementos involucrados en la migración. Esto fue el punto de partida para la extensión del mismo, ya que, a partir del esquema definido por PKP, fue posible



pensar y desarrollar un esquema más complejo, que tenga en cuenta el proceso editorial del artículo, mas no deje de ser extensible para futuros desarrollos. Luego, fue necesario entender cómo manipula la información OJS dentro de la misma aplicación (a través de los DAO, como se marcó en el capítulo 3), para así poder integrar los nuevos metadatos necesarios, correspondientes al envío en el proceso de exportación.

Realizado el desarrollo del nuevo plugin, se llevaron a cabo pruebas de campo en un ambiente controlado en donde se involucraron 2 instancias OJS distintas y cada una de ellas contenía una revista para realizar la prueba de migración. Se determinó que, para que este nuevo plugin funcione correctamente, es necesario que ambas instalaciones de OJS cumplan ciertos requisitos como por ejemplo que tengan una versión específica y se importen los usuarios y formularios personalizados. Se puede ver que el nuevo plugin abarca todas las etapas de los envíos (*submission, review, copyediting, production*), tanto así como los que fueron rechazados o publicados.

## Ventajas

Como se ha expresado a lo largo de este proyecto, haber determinado como alternativa de solución un desarrollo a partir de la extensión de un plugin existente, el cual está desarrollado por los mismos creadores de OJS, y seguir su estructura implica una serie de ventajas y muchas de estas se han mencionado durante la tesis.

En primer lugar, se puede remarcar que un gran beneficio que se da en este desarrollo es la capacidad de mantener la integridad de los datos, que surge a partir del re-uso de los DAO implementados por PKP a la hora de obtener y almacenar información. Esto implica que gran parte del desarrollo no necesita tener en cuenta dichas implementaciones y reglas de negocio propias de PKP y del proceso editorial en sí, lo que implica una gran ayuda a la hora de resolver, por ejemplo, la creación de un envío desde cero, el estado en el que se encuentran las revisiones, la modificación de etapas de un envío y el agregado de discusiones, participantes, etc.

En segundo lugar, se puede mencionar la fácil integración que tiene con OJS, nuevamente, al ser una extensión de un plugin creado por el grupo PKP, con el solo hecho de reemplazar el código del viejo plugin por el nuevo, quedará instalado para generar migraciones de envíos. Conjuntamente, dada la naturaleza del plugin, no es necesaria ninguna herramienta de terceros

que tenga que intervenir en el proceso de exportación; este se realiza exclusivamente dentro de la misma instalación de OJS. Si se hubiera optado por otro tipo de solución que implicase conectar OJS a otro sistema o tecnología, esta se tornaría más compleja de integrar, verificar el funcionamiento y mantener. Además, influye el hecho de que un usuario de OJS con cierta experiencia conoce el funcionamiento del plugin original, por lo que no necesita ningún tipo de instrucción nueva para utilizarlo.

En tercer lugar, otra gran ventaja de este desarrollo se relaciona con cómo fue pensado el proyecto. Se planteó como una mejora del plugin *Native*, por ende, a pesar de que se hayan agregado nuevas funcionalidades para incluir más metadatos en el proceso de exportación, se preservan todas sus funcionalidades anteriores que son de gran utilidad. Por ejemplo, en el capítulo 4 se vio que en ciertos casos, como en el de envío rechazado o publicado, ese manejo de estados del artículo ya estaba implementado por el plugin original, lo que implica menos aspectos a tener en cuenta para agregar en la nueva extensión y aliviana el desarrollo.

En cuarto lugar, se destaca la simpleza que implica el uso del plugin. Con esta extensión, en caso de que los administradores de un portal deseen generar una migración a otra instalación OJS, pueden realizar una migración modulada, delegando, si quisieran, distintos números a cada uno de los responsables del portal. Cada uno de ellos puede generar la exportación de cada número y controlarla, ya que vimos que el plugin provee la posibilidad de exportar/importar más de un artículo a la vez, todos agrupados en un XML. Además, hay que tener en consideración que el hecho de que el plugin provea una migración automática evita los errores que puedan ocurrir en caso de replicar el proceso manualmente (lo que sería una tarea muy difícil para un gran número de envíos), especialmente si se hace foco en todos los idiomas en metadatos que pueden involucrarse durante el envío de un artículo.

En quinto lugar, se puede mencionar la velocidad con la que se realizan las migraciones; el tiempo es prácticamente despreciable, aún si se realiza envío por envío. Aunque en general se recomienda realizar la exportación/importación de paquetes de envíos, como números completos. Por ejemplo, en el caso de que un envío se quisiera migrar manualmente a otro OJS, esto implicaría:

- obtener todos sus archivos de todas las etapas, lo que involucraría revisar a qué etapa corresponde cada uno;
- crear el envío en la nueva instalación;

- asignarle sus participantes con sus respectivos roles y si no los tuviera en la instalación, agregarlos;
- crear las discusiones de cada etapa las que tendrían que ser modificadas, no a través de la aplicación, sino de la base de datos, ya que las fechas de las mismas no corresponderían al envío original;
- crear todas las rondas de revisión con todas sus revisiones (con sus archivos y formularios) y revisores asignados además de modificarlas con el mismo criterio que se planteó para las discusiones (teniendo en cuenta que para las revisiones el manejo de fechas es significativamente más complejo);
- asignar todos los archivos correspondientes a cada etapa;
- dejar el artículo en la etapa que le corresponde.

Este proceso corresponde a un solo envío, si se extrapola a un número completo o incluso, a una revista, el trabajo podría suponer semanas. Con la nueva extensión, esto se realiza en minutos.

Por último, resulta importante hacer hincapié en que, si bien la problemática que busca solucionar este desarrollo es algo que se ha planteando a lo largo de estos años, no existe ningún tipo de solución ni de PKP ni de la misma comunidad de OJS. Persiste a lo largo del tiempo y de las distintas versiones de OJS y que PKP no ha resuelto. Por estas razones, este proyecto podría ser una parte representativa de la solución que requiere la comunidad; es un primer paso hacia mejorar recursos para lograr la exportación de envíos, con todo su proceso editorial, brindar una opción simple para la integración de instalaciones OJS y motivar la estandarización de datos de los artículos y su proceso editorial.

## Desventajas

Si bien se considera que este proyecto genera un gran aporte a los usuarios de portales de revista en OJS y, además, brinda una serie de ventajas a la hora de migrar envíos, se debe dejar en claro que tiene puntos débiles que pueden ser mejorados.

En primer lugar, se presenta la problemática del versionado. Como se mencionó en el capítulo 3, esta extensión fue desarrollada para la versión 3.2.1, que era una de las últimas estables de OJS. Esto implica una limitación de uso del plugin, ya que solamente podrá ser usado para instalaciones en dicha versión; esto se debe a que cada salto de versión 3.x afecta al esquema

de datos y a la forma de obtenerlos (esto sucede en mayor medida con las versiones 2.4.x, que son aún más antiguas). Entonces, en caso de que una instalación de OJS quiera hacer uso del nuevo plugin deberá realizar un *upgrade* a la versión 3.2.1. Este proceso de *upgrade* puede tornarse complejo en función de la versión de OJS que posea la institución, particularmente si una entidad desea actualizar a partir de la versión 2.4.x, ya que involucra una gran cantidad de cambios en la aplicación y su esquema de datos.

En segundo lugar, se puede mencionar que el plugin no incluye un sistema de control de errores, por ende, en caso de que ocurra alguno durante el proceso de exportación, el usuario no tendrá acceso a un reporte de las causas. Asimismo, si ocurre un error durante el proceso de importación, el envío quedará importado de forma incompleta (este tendrá toda la información importada hasta el momento en el que ocurra el error). Por ejemplo, se puede dar el caso de que un usuario ejecuta la importación de un envío en el que en la etapa de revisión se utilizaron ciertos formularios personalizados, sin embargo, dicho usuario omite la importación de estos formularios lo que conlleva a un error, ya que el OJS destino no los posee. Este error se generaría en la importación de la etapa de revisión (la cual es la segunda etapa de un envío), por lo que toda la importación de las etapas subsecuentes no se llevaría a cabo.

En tercer lugar, en este proyecto se hizo mucho énfasis en que, para lograr el correcto funcionamiento del nuevo plugin, previamente se debe tener en la instalación OJS destino los mismos usuarios, roles y formularios personalizados de la instalación origen. Este punto no está controlado por el plugin y es un proceso que debe realizar el usuario manualmente. Como se mencionó anteriormente en esta sección, la ausencia de los elementos mencionados puede ocasionar un error en la importación de un envío.

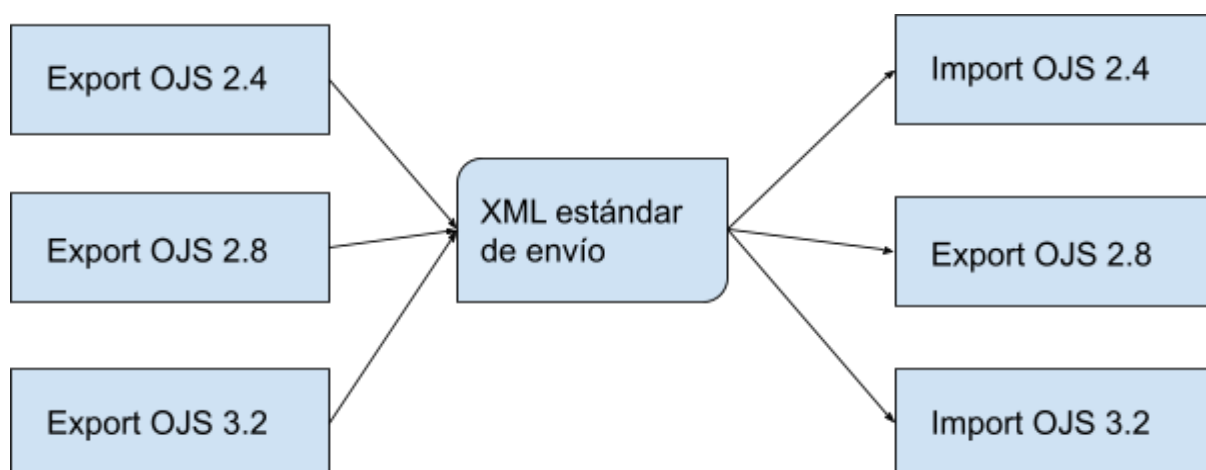
En cuarto lugar, debido a algunas implementaciones del plugin *Native* original, asociada a los archivos a exportar y al hecho de no querer modificar el código fuente sino extenderlo, han aparecido soluciones forzadas dentro del proyecto. Un ejemplo de esto está relacionado con los archivos en la etapa de revisión. Si bien estos están incluidos en el proceso de exportación original, estos sólo poseen a qué etapa pertenecen y vimos que el XML inicial no cuenta con ninguna estructura que delimite las rondas dentro de la etapa de revisión, por ende, el desarrollo cuenta con este tipo de soluciones. Para solucionar el problema de los archivos de revisión, el proceso de exportación almacena la cantidad de materiales que posee cada ronda (en la nueva estructura de *rounds*) y, luego, el proceso de exportación recorre los archivos que pertenezcan a la etapa de revisión y los agrega en base a la cantidad que diga cada ronda. Como

se puede ver, esta no es una solución idónea, ya que está limitada al orden en que se guardan los archivos y al orden de las rondas. Una solución adecuada sería que cada ronda guarde los archivos que le competen, pero esto implicaría tener el archivo duplicado, ya que fue prioridad no alterar el comportamiento original de *Native*, sino extenderlo. Además, este manejo de archivos por parte de *Native* también afectó negativamente a la exportación de los archivos de las *reviewAssignments*. No se pudieron incluir en la exportación ya que, si bien se sabe cuántos *reviewAssignments* hay en cada ronda, los archivos que pertenecen a esta etapa de la revisión no pueden ser delimitados por un orden (ya que las *reviewAssignments* varían en el tiempo dependiendo de sus revisores y sus archivos también), en consecuencia, la solución implicaba modificar el código original de *Native* para incluirlos.

## Trabajos a futuro

A partir de las desventajas planteadas, pueden surgir desarrollos que sirvan para hacerles frente, lo que implicaría una serie de mejoras para el plugin. Se mencionará cada desarrollo en el mismo orden que se han mencionado las desventajas respectivamente.

Una solución que se podría llegar a implementar para la problemática del versionado sería un desarrollo basado en adaptadores. Esto implicaría pensar ambas partes como procesos independientes que varíen en base a la versión de la instalación de OJS, con el objetivo de crear y consumir siempre el mismo esquema XML (esto daría lugar a una idea de estándar). Con una serie de adaptadores creados para cada versión de OJS, tanto en la etapa de exportación como la de importación, sería factible llegar a pensar que cualquier instalación de OJS podría generar un archivo XML representando al envío y cualquier otra instalación podría consumirlo.



*Grafico 3: Gráfico que muestra un posible escenario en donde intervienen diferentes versiones del plugin para exportar e importar envíos. Fuente: elaboración propia.*

Esta solución abre un nuevo escenario en el que se podría llegar a lograr la interoperabilidad entre distintos OJS. Un caso podría ser que cada OJS posea sus adaptadores correspondientes para lograr una exportación de un envío, lo que crearía un XML que seguiría un estándar de interoperabilidad entre los mismos; a esto se le puede sumar que cada instalación cuente con la posibilidad de generar una conexión con otra (por ejemplo por HTTP) por la que se pudieran intercambiar dichos XML y lograr migraciones online. Sólo el hecho de posibilitar que cualquier portal pueda comunicarse con otro para compartir envíos dentro de la misma aplicación es un gran aporte a la de PKP. Lograr esta interoperabilidad puede generar lazos, no sólo entre portales de un mismo país, sino conectar portales de distintos países a lo largo del mundo.

Al hacer foco en el problema del control errores, indudablemente el plugin necesitaría tener implementado un manejador de errores, tanto en la parte de exportación como en la de importación, para que el usuario tenga una respuesta adecuada del sistema y pueda responder a dichos errores.

En este sentido, resulta importante tener en cuenta qué medidas puede tomar el plugin ante la presencia de algún error en el proceso de importación. Por ejemplo, actualmente al no haber ningún control de error, el plugin genera una importación de un envío hasta que algún error se presenta, detiene el proceso y crea una importación incompleta. Si bien puede que esta política sea deseable para el usuario, con una serie de modificaciones en el código (como por ejemplo la implementación de checkpoints) el plugin podría generar una importación del estilo *best-effort* acompañado de un *feedback* que indique todos los elementos que no pudieron ser importados y el motivo de cada uno. Es decir, se puede dar el caso de que, durante el proceso de importación de un envío, ocurra un error al importar la etapa de revisión porque previamente no se importó el usuario revisor que interviene en la misma. Con la política planteada, el plugin importaría todo lo que puede del artículo y, para el caso del error, informaría al usuario que no se ha podido importar las revisiones debido a que el OJS no posee dicho revisor.

Otra medida puede ser una importación del estilo “todo o nada” donde, en caso de que ocurra un error, el plugin realizaría un *rollback* de los datos del envío que se quiera importar, lo que evitaría cualquier importación incompleta. Esta política también debería retornar un listado de errores. En este caso, si se toma el mismo ejemplo mencionado para la política anterior, al haber

ocurrido un error en la etapa de revisión, el plugin desharía toda la importación realizada hasta el momento e informaría al usuario el motivo. Además, llevar a cabo este tipo de soluciones no debería requerir una excesiva demanda de tiempo, dado que sólo implicaría analizar cuáles serían las causas que podrían generar errores en el proceso de importación y crear un plan de contingencia que concretaría el objetivo planteado al principio de este párrafo.

De la mano de este análisis de errores, se puede evaluar el hecho de que el nuevo plugin necesita que la instalación OJS destino posea la misma configuración de usuario, de roles y de formularios personalizados que la origen. Al ser un proceso que depende del usuario, esto puede resultar tedioso y propenso a fallos. Esto podría evitarse incorporando algún módulo de chequeo dentro del plugin que evalúe si previamente todas las condiciones (usuarios, roles, formularios personalizados) están dadas para realizar una importación de un envío. De igual forma, en caso de que alguna faltase, se podría incluir algún módulo de importación para cada una de las precondiciones necesarias. Por ejemplo, en caso de que haya usuarios faltantes, el nuevo plugin podría proponer ejecutar, o hasta incorporar en su desarrollo, algún otro plugin existente para la importación de los mismos (en este caso podría ser el *import/export Users* de PKP) y así solucionar el problema. Esto puede ser replicado para roles y formularios personalizados. Sumado a esto, dado que actualmente OJS provee un plugin de exportación de usuarios, la base de su desarrollo podría ser extendida (para que incluya todos los elementos antes mencionados) e incorporada al plugin de exportación, lo que permitiría una rápida solución frente a la problemática planteada.

Una posible propuesta para solventar las soluciones forzadas, puede ser sugerir un cambio en el código fuente del plugin *Native* de PKP y reestructurar el XSD que tiene creado. A lo largo del proyecto se ha hecho hincapié reiteradas veces sobre la importancia que tuvo extender el plugin *Native* y evitar realizarle alguna modificación que implicase cambiar su funcionamiento original. Si bien todo esto puede ser ignorado para un desarrollo en particular, el objetivo no es limitarse a un desarrollo personalizado para una(s) instalación(nes) de OJS, sino plantear un esquema válido y que sea legitimado tanto por PKP como por la comunidad. Dadas estas circunstancias, este tipo de solución tiende a ser más compleja; sin embargo, ya se ha hablado de plantear un esquema de interoperabilidad, por lo que incluir en esta idea un cambio de base en el XSD de *Native* puede ser una consideración.

## Conclusión

Dado que la calidad de las revistas científicas es de suma importancia, ya que este parámetro las posiciona de diferente manera ante la comunidad científica y el mundo, las entidades que las manejan deben mantener rigurosidad y transparencia a lo largo de su proceso editorial. Si bien dicho proceso tiene definido cierto esquema de trabajo, el manejo de las revistas puede variar en la forma de ejecutar el proceso editorial según sus posibilidades, ya sea por una cuestión estructural o de recursos (económicos, humanos, etc). Sumado a esto, resulta importante que esas entidades mantengan registrados todos los trabajos que pasan por sus propios procesos editoriales, ya que esta información resulta valiosa, tanto para mostrar transparencia como para utilizarla de *feedback* con el objetivo de mejorar la calidad de los mismos procesos editoriales. Asimismo, existen herramientas, como OJS, que permiten la gestión de portales de revistas y de procesos editoriales (además de una abundante cantidad de funcionalidades), por lo que muchos equipos optan por este software. Dentro de su vasto repertorio de utilidades, OJS provee la posibilidad de generar exportaciones e importaciones de artículos entre diferentes instalaciones. Sin embargo, esta funcionalidad, como ya se mencionó, está orientada solamente a artículos, sin tomar en cuenta su proceso; por consiguiente, si se quisiera migrar una revista de una instalación OJS a otra, toda esa información valuable se “pierde” para la instalación OJS destino; como ya se ha mencionado en este proyecto, la necesidad de migrar revistas entre diferentes entidades es una situación verosímil.

Nacido de una idea y de una necesidad planteada por PREBI-SEDICI, este proyecto propone una extensión a lo que ya aporta PKP en su plugin de exportación e importación. A partir de la estructura trazada por PKP, se amplió la complejidad de dicho plugin por lo que ahora este provee la posibilidad de migrar un artículo y gran parte de su proceso editorial hacia otra revista en otra instalación de OJS. Esto es posible siempre y cuando se respeten los requisitos del plugin que se han mencionado, a lo largo de la tesis. Este proyecto plantea una solución puntual a una necesidad amplia y compleja. El objetivo trasciende dicha solución y busca incentivar un desarrollo que en el futuro pueda ser utilizado por la gran mayoría de las instalaciones OJS y así, lograr una facilidad para las mismas, independientemente de su versión, proceso editorial, lenguaje, etc.; además, dentro de lo posible, motivar una comunicación de OJS a lo largo del mundo, y así poder intercambiar ideas, metodologías e información.



## Referencias

1. Abadal, E. y Rius Alcaraz, L. (2008). Revistas científicas de las universidades españolas: acciones básicas para aumentar su difusión e impacto. *Revista española de Documentación Científica*, 31(2), 240-260. <http://dx.doi.org/10.3989/redc.2008.v31.i2.427>
2. Ahmar, A. S., Kurniasih, N., Irawan, D. E., Sutiksno, D. U., Napitupulu, D., Setiawan, M. I., ... y Abraham, J. (2018). Lecturers' understanding on indexing databases of SINTA, DOAJ, Google Scholar, SCOPUS, and Web of Science: A study of Indonesians. *Journal of Physics: Conference Serie*, 954, 012026. <https://iopscience.iop.org/article/10.1088/1742-6596/954/1/012026>
3. Alperin, J. P., Stranack, K. y Garnett, A. (2016). On the peripheries of scholarly infrastructure: a look at the journals using open journal systems. *21st International Conference on Science and Technology Indicators-STI*. Valencia (España), 14-16 de Septiembre. <http://ocs.editorial.upv.es/index.php/STI2016/STI2016/paper/viewFile/4543/2327>
4. Aretio, L. G. (2014). OJS y DOI, apuestas por la calidad de las revistas científicas. *Revista Iberoamericana de Educación a Distancia*, 17(2), 9-13. <https://doi.org/10.5944/ried.17.2.12675>
5. Artini, M., Atzori, C., Bardi, A., La Bruzzo, S., Manghi, P. y Mannocci, A. (2015). The OpenAIRE literature broker service for institutional repositories. *D-Lib Magazine*, 21(11/12), 1. <https://dx.doi.org/10.1045/november2015-artini>
6. Barrueco, J. M. y Subirats-Coll, I. (2003). Open Archives Initiative Protocol for Metadata Harvesting (OAI-PMH): descripción, funciones y aplicación de un protocolo. *El Profesional de la Información*, 12(2), 99-106.
7. Becerril Garcia, A., Lozano Espinosa, R. y Molina Espinosa, J. M. (2016). Enfoque semántico para el descubrimiento de recursos sensible al contexto sobre contenidos académicos estructurados con OAI-PMH. *Computación y Sistemas*, 20(1), 127-142. <https://doi.org/10.13053/cys-20-1-2189>
8. Bosch X., Alfonso F. y Bermejo J. (2002). ¿Por qué se ha rechazado nuestro artículo? *Revista Española de Cardiología*, 55 (7), 114-115.
9. Deka, D. (2007). OAI-PMH: A tool for metadata harvesting and federated search. En *PLANNER 2007*. INFLIBNET Centre. <http://ir.inflibnet.ac.in/handle/1944/1360>
10. De Giusti, M., Villarreal, G. L., Terruzzi, F. A., Oviedo, N. F. y Lira, A. J. (2013). Interoperabilidad entre el Repositorio Institucional y servicios en línea en la Universidad Nacional de La Plata. *PKP International Scholarly Publishing Conferences (Mexico, 2013)*.

<http://sedici.unlp.edu.ar/handle/10915/27406>

11. De Giusti, M. R., Adorno, F. G. y Lira, A. J. (2014). Repositorios DSpace con múltiples contextos OAI-PMH. Trabajo presentado en *Conferência Internacional Acesso Aberto, Preservação Digital, Interoperabilidade, Visibilidade e Dados Científicos - BIREDIAL 2014 (Brasil, 2014)*. <https://hdl.handle.net/10915/41628>
12. De Giusti, M., García, D., Manzur, E., Villarreal, G. L. y Folegatto, L. E. (2018). Importación de artículos en OJS desde Dspace. En *VIII Conferencia Internacional sobre Bibliotecas y Repositorios Digitales BIREDIAL-ISTEC (Lima)*. <http://sedici.unlp.edu.ar/handle/10915/70581>
13. Guzmán González, R. y Díaz, A. L. (2019). La incidencia de la tecnología en la edición de revistas científicas: el uso de Open Journal System (OJS) en la Universidad Nacional de Salta. En *XXI° Congreso de la Red de Carreras de Comunicación Social y Periodismo*. Escuela de Ciencias de la Comunicación, Facultad de Humanidades (UNSa), Salta, Argentina, 16-18 de octubre.
14. Houssos N., Stamatis K., Banos V., Kapidakis S., Garoufallou E. y Koulouris A. (2011). Implementing Enhanced OAI-PMH Requirements for Europeana. En: S. Gradmann, F. Borri., C. Meghini y H. Schuldt (eds), *Research and Advanced Technology for Digital Libraries*. TPD 2011. Lecture Notes in Computer Science, vol 6966. Springer, Berlin, Heidelberg.
15. Jiménez, J. S. y Castañeda, M. A. H. (2003). Algunas consideraciones sobre la evaluación de la calidad de las revistas. *Revista de Enfermería IMSS*, 11(1), 1-3.
16. Lujano, I. (2017). Collaboration in Global South Open Access landscape: A perspective of DOAJ in Latin America. En *Public Knowledge Project (PKP) 2017*. <https://doi.org/10.5446/51336#t=00:01,04:49>
17. Rodriguez, M. A. y Bollen, J. (2008). An algorithm to determine peer-reviewers. En *Proceedings of the 17th ACM conference on Information and knowledge management (CIKM '08)* (pp. 319–328). Association for Computing Machinery (USA). <https://doi.org/10.1145/1458082.1458127>
18. Miyahira, J. M. (1995). El arbitraje editorial en las revistas médicas. *Revista Médica Herediana*, 6 (3), 106.
19. Pasteur, L. (2020). La importancia del DOI y el formato HTML en la difusión del conocimiento médico y científico actual. *Revista Mexicana de Endocrinología, Metabolismo & Nutrición*, 7, 5-9. <https://doi.org/10.24875/RME.20000002>
20. [PKP20] PKP - Public Knowledge Project Open Journal Systems (n.d.). Recuperado el 20 de Agosto, 2020, de <https://pkp.sfu.ca/ojs/>
21. Polanco-Cortés, J. y Garro Acón, M. (2013). Propuesta de modelo centroamericano para la creación de portales nacionales de revistas a través de las universidades estatales. En *III Conferencia Internacional de "Acceso abierto, preservación digital y datos científicos", III Conferencia Bibliotecas y Repositorios Digitales de América Latina (BIREDIAL '13), VIII Simposio Internacional de Bibliotecas*

*Digitales (SIBD '13)*, Ciudad de la Investigación, Universidad de Costa Rica, 15-17 de octubre.  
<http://hdl.handle.net/10760/20568>

22. Rojas, M. y Rivera, S. (2011). *Guía de Buenas Prácticas para Revistas Académicas de Acceso Abierto*. ONG Derechos Digitales.
23. Hernandez, R. (2017). Proceso editorial de una revista científica: cumpliendo con los requisitos de publicación. *Revista Peruana De Psicología Y Trabajo Social*, 4(1), 77-84.
24. Rozemblum, C., Unzurrunzaga, C., Banzato, G. y Pucacco, C. (2015). Calidad editorial y calidad científica en los parámetros para inclusión de revistas científicas en bases de datos en Acceso Abierto y comerciales. *Palabra Clave (La Plata)*, 4(2), 64-80.
25. Schirrwagen, J. y Baglioni, M. (2018). *OpenAIRE Guidelines for institutional and thematic repository managers 4.0*. Zenodo. <https://doi.org/10.5281/ZENODO.1299202>
26. Schonhaut Berman, L., Millán Klusse, T. y Podestá López, L. (2017). Revisión por pares: evidencias y desafíos. *Revista Chilena de Pediatría*, 88(5), 577-581.  
<https://dx.doi.org/10.4067/S0370-41062017000500001>
27. Thompson, H. S., Mendelsohn, N., Beech, D. y Maloney, M. (2009). W3C XML schema definition language (XSD) 1.1 part 1: Structures. The World Wide Web Consortium (W3C), W3C Working Draft Dec. Recuperado de <https://www.w3.org/TR/2009/WD-xmlschema11-2-20090130/>
28. Rivero Torres, C., Velepucha Sánchez, M., Rosillo Suárez, A., & Suárez Mella, R. (2019). Beneficios de la Plataforma Open Journal Systems para las revistas digitales UTM. *Revista San Gregorio*, 0(28).  
[doi:http://dx.doi.org/10.36097/rsan.v0i28.787](http://dx.doi.org/10.36097/rsan.v0i28.787)
29. Torres, C. R., Rivadeneira, J. G., Sánchez, M. V., Vences, J. B., Briones, G. S. y García, L. A. M. (2020). Plataforma Open Journal Systems en la gestión de publicaciones científicas. *Revista Electrónica Cooperación Universidad Sociedad*, 5(2), 1-6. <https://doi.org/10.33936/recus.v5i2.1866>
30. Villarreal, G. L., Manzur, E., García, D. y De Giusti, M. R. (2019). Integración de revistas desde repositorios institucionales hacia Open Journal Systems. En *IX Conferência Internacional sobre Bibliotecas e Repositórios Digitais da América Latina (BIREDIAL-ISTEC'19)(São Paulo, Brasil)*, 30 de julio- 2 de agosto.  
<https://hdl.handle.net/10915/7887>

## Anexo

XML native.xsd actual de OJS

```
<?xml version="1.0"?>
schema xmlns="http://www.w3.org/2001/XMLSchema" targetNamespace="http://pkp.sfu.ca"
xmlns:pkp="http://pkp.sfu.ca" elementFormDefault="qualified">
<include schemaLocation="../../lib/pkp/plugins/importexport/native/pkp-native.xsd" />
<element name="article" substitutionGroup="pkp:submission" type="pkp:articleInfo">
</element>
  <complexType name="articleInfo">
    <complexContent>
      <extension base="pkp:submission">
        <attribute name="stage" use="required">
          <simpleType>
            <restriction base="string">
              <enumeration value="submission" />
              <enumeration value="externalReview" />
              <enumeration value="editorial" />
              <enumeration value="production" />
            </restriction>
          </simpleType>
        </attribute>
      </extension>
    </complexContent>
  </complexType>
</element>
<element name="articles" substitutionGroup="pkp:submissions" />
<element name="citation" type="string" />
<element name="publication" substitutionGroup="pkp:pkppublication">
  <complexType>
    <complexContent>
      <extension base="pkp:pkppublication">
        <sequence>
          <element ref="pkp:issue_identification" minOccurs="0"
maxOccurs="1" />
          <element maxOccurs="1" minOccurs="0" name="pages"
type="string" />
          <element ref="pkp:covers" minOccurs="0" maxOccurs="1" />
          <element maxOccurs="1" minOccurs="0" name="issuelid"
type="int" />
          <element ref="pkp:citations" minOccurs="0"
maxOccurs="unbounded" />
        </sequence>
        <attribute name="section_ref" type="string" use="required" />
      </extension>
    </complexContent>
  </complexType>
</element>
```

```
                <attribute name="seq" type="int" />
                <attribute name="access_status" type="int" />
            </extension>
        </complexContent>
    </complexType>
</element>

<element name="article_galley" substitutionGroup="pkp:representation">
    <complexType>
        <complexContent>
            <extension base="pkp:representation">
                <attribute name="approved" type="boolean"/>
                <attribute name="galley_type" type="string"/>
            </extension>
        </complexContent>
    </complexType>
</element>

<element name="artwork_file" substitutionGroup="pkp:submission_file">
    <complexType>
        <complexContent>
            <extension base="pkp:submission_file">
                <sequence>
                    <element maxOccurs="1" minOccurs="0" name="caption"
                        type="string" />
                    <element maxOccurs="1" minOccurs="0" name="credit"
                        type="string" />
                    <element maxOccurs="1" minOccurs="0" name="copyright_owner"
                        type="string" />
                    <element maxOccurs="1" minOccurs="0"
                        name="copyright_owner_contact" type="string" />
                    <element maxOccurs="1" minOccurs="0"
                        name="permission_terms" type="string" />
                </sequence>
            </extension>
        </complexContent>
    </complexType>
</element>

<element name="supplementary_file" substitutionGroup="pkp:submission_file">
    <complexType>
        <complexContent>
            <extension base="pkp:submission_file">
                <sequence>
                    <element name="creator" type="pkp:localizedNode" />
                </sequence>
            </extension>
        </complexContent>
    </complexType>
</element>
```

```
minOccurs="0" maxOccurs="unbounded" />
    <element name="subject" type="pkp:localizedNode"
minOccurs="0" maxOccurs="unbounded" />
    <element name="description" type="pkp:localizedNode"
minOccurs="0" maxOccurs="unbounded" />
    <element name="publisher" type="pkp:localizedNode"
minOccurs="0" maxOccurs="unbounded" />
    <element name="sponsor" type="pkp:localizedNode"
minOccurs="0" maxOccurs="unbounded" />
    <element name="date_created" type="date" minOccurs="0"
maxOccurs="1" />
    <element name="source" type="pkp:localizedNode" minOccurs="0"
maxOccurs="unbounded" />
    <element name="language" type="string" minOccurs="0"
maxOccurs="1" />
    </sequence>
  </extension>
</complexType>
</element>

<element name="issues">
  <complexType>
    <sequence>
      <element ref="pkp:issue" maxOccurs="unbounded" />
    </sequence>
  </complexType>
</element>

<complexType name="issue">
  <sequence>
    <element ref="pkp:id" minOccurs="0" maxOccurs="unbounded" />
    <element name="description" type="pkp:localizedNode" minOccurs="0"
maxOccurs="unbounded" />
    <element ref="pkp:issue_identification" minOccurs="1" maxOccurs="1" />

    <element name="date_published" type="date" minOccurs="0" maxOccurs="1" />
    <element name="date_notified" type="date" minOccurs="0" maxOccurs="1" />
    <element name="last_modified" type="date" minOccurs="0" maxOccurs="1" />
    <element name="open_access_date" type="date" minOccurs="0" maxOccurs="1" />

    <element ref="pkp:sections" minOccurs="0" maxOccurs="1" />
    <element ref="pkp:covers" minOccurs="0" maxOccurs="1" />
    <element ref="pkp:issue_galleys" minOccurs="0" maxOccurs="1" />
    <element ref="pkp:articles" minOccurs="1" maxOccurs="1" />
  </sequence>
  <attribute name="journal_id" type="int" use="optional" />

```

```
<attribute name="published" type="int" use="optional" />
<attribute name="current" type="int" use="optional" />
<attribute name="access_status" type="int" use="optional" />
<attribute name="url_path" type="string" use="optional" />
</complexType>

<element name="issue_identification">
  <complexType>
    <sequence>
      <element name="volume" type="int" minOccurs="0" maxOccurs="1" />
      <element name="number" type="string" minOccurs="0" maxOccurs="1" />
      <element name="year" type="int" minOccurs="0" maxOccurs="1" />
      <element name="title" type="pkp:localizedNode" minOccurs="0"
maxOccurs="unbounded" />
    </sequence>
  </complexType>
</element>

<complexType name="issue_galley">
  <sequence>
    <element name="label" type="string" minOccurs="1" maxOccurs="1" />
    <element ref="pkp:id" minOccurs="0" maxOccurs="unbounded" />
    <element ref="pkp:issue_file" minOccurs="0" maxOccurs="1" />
  </sequence>
  <attribute name="locale" type="string" />
</complexType>

<complexType name="issue_file">
  <sequence>
    <element name="file_name" type="string" minOccurs="1" maxOccurs="1" />
    <element name="file_type" type="string" minOccurs="1" maxOccurs="1" />
    <element name="file_size" type="int" minOccurs="1" maxOccurs="1" />
    <element name="content_type" type="int" minOccurs="1" maxOccurs="1" />
    <element name="original_file_name" type="string" minOccurs="1" maxOccurs="1" />
    <element name="date_uploaded" type="date" minOccurs="1" maxOccurs="1" />
    <element name="date_modified" type="date" minOccurs="1" maxOccurs="1" />
    <element name="embed" type="pkp:embed" />
  </sequence>
</complexType>

<element name="covers">
  <complexType>
    <sequence>
      <element ref="pkp:cover" minOccurs="1" maxOccurs="unbounded" />
    </sequence>
  </complexType>
</element>
```

```
<element name="cover">
  <complexType>
    <sequence>
      <element name="cover_image" type="string" minOccurs="1" maxOccurs="1" />
      <element name="cover_image_alt_text" type="string" minOccurs="1"
maxOccurs="1" />
      <element name="embed" type="pkp:embed" />
    </sequence>
    <attribute name="locale" type="string" />
  </complexType>
</element>

<element name="issue_galleys">
  <complexType>
    <sequence>
      <element ref="pkp:issue_galley" minOccurs="0" maxOccurs="unbounded" />
    </sequence>
  </complexType>
</element>

<element name="sections">
  <complexType>
    <sequence>
      <element ref="pkp:section" maxOccurs="unbounded" />
    </sequence>
  </complexType>
</element>

<complexType name="section">
  <sequence>
    <element ref="pkp:id" minOccurs="0" maxOccurs="unbounded" />
    <element name="abbrev" type="pkp:localizedNode" minOccurs="0" maxOccurs="unbounded"

    <element name="policy" type="pkp:localizedNode" minOccurs="0" maxOccurs="unbounded"

    <element name="title" type="pkp:localizedNode" minOccurs="0" maxOccurs="unbounded" />
  </sequence>
  <attribute name="ref" type="string" />
  <attribute name="review_form_id" type="int" use="optional" />
  <attribute name="seq" type="string" />
  <attribute name="editor_restricted" type="int" />
  <attribute name="meta_indexed" type="int" />
  <attribute name="meta_reviewed" type="int" />
  <attribute name="abstracts_not_required" type="int" />
  <attribute name="hide_title" type="int" />
  <attribute name="hide_author" type="int" />
  <attribute name="abstract_word_count" type="int" />
</complexType>
```



```
<element name="issue" type="pkp:issue" />

<element name="issue_galley" type="pkp:issue_galley" />

<element name="issue_file" type="pkp:issue_file" />

<element name="section" type="pkp:section" />

schema>
```

## XML native.xsd extendido

```
xml version="1.0"?>
--
plugins/importexport/native/native.xsd
Copyright (c) 2014-2020 Simon Fraser University
Copyright (c) 2003-2020 John Willinsky
Distributed under the GNU GPL v3. For full terms see the file docs/COPYING.

Schema describing native XML import/export elements specific to OJS
>
schema xmlns="http://www.w3.org/2001/XMLSchema" targetNamespace="http://pkp.sfu.ca"
xmlns:pkp="http://pkp.sfu.ca" elementFormDefault="qualified">

<redefine schemaLocation="./native_old.xsd" >
  <complexType name="articleInfo">
    <complexContent>
      <extension base="articleInfo">
        <sequence>
```

```
<element ref="pkp:participants"></element>
<element ref="pkp:stages"></element>
</sequence>
</extension>
</complexContent>
</complexType>
</redefine>
element name="participants">
  <complexType>
    <sequence>
      <element ref="pkp:participant" minOccurs="1" maxOccurs="unbounded" />
    </sequence>
  </complexType>
</element>
element name="participant">
  <complexType>
    <attribute name="mail" type="string"/>
    <attribute name="user_group_ref" type="string"/>
  </complexType>
</element>
element name="stages">
  <complexType>
    <sequence>
      <element ref="pkp:stage" minOccurs="5" maxOccurs="5" />
    </sequence>
  </complexType>
</element>
element name="stage">
  <complexType>
    <sequence>
```

```
<element ref="pkp:queries" />
<element ref="pkp:rounds" minOccurs="0" maxOccurs="1" />
</sequence>
<attribute name="id" />
<attribute name="name" type="string" />
</complexType>
</element>

<element name="rounds">
  <complexType>
    <sequence>
      <element ref="pkp:round" minOccurs="0" maxOccurs="unbounded" />
    </sequence>
  </complexType>
</element>

<element name="round">
  <complexType>
    <sequence>
      <element ref="pkp:reviewAssignment" minOccurs="0" maxOccurs="unbounded" />
      <element ref="pkp:file" minOccurs="0" maxOccurs="unbounded" />
    </sequence>
  </complexType>
</element>

<element name="file">
  <complexType>
    <sequence>
      <element ref="pkp:name"/>
      <element ref="pkp:embed" minOccurs="0"/>
    </sequence>
    <attribute name="id" />
    <attribute name="number" />
  </complexType>
</element>
```

```
<attribute name="stage" />
<attribute name="source" />
<attribute name="genre" />
<attribute name="filename" />
<attribute name="viewable" />
<attribute name="date_uploaded" />
<attribute name="date_modified" />
<attribute name="filesize" />
<attribute name="filetype" />
<attribute name="uploader" />
/complexType>
element>

element name="name">
complexType mixed="true">
<attribute name="locale" />
/complexType>
element>

element name="embed">
complexType mixed="true">
<attribute name="encoding" />
/complexType>
element>

element name="reviewAssignment">
complexType>
<sequence>
<element ref="pkp:form" minOccurs="1" />
</sequence>
<attribute name="reviewer" />
<attribute name="method" />
<attribute name="round" />
```

```
<attribute name="unconsidered" />
<attribute name="date_rated" />
<attribute name="last_modified" />
<attribute name="date_assigned" />
<attribute name="date_notified" />
<attribute name="date_confirmed" />
<attribute name="date_completed" />
<attribute name="date_acknowledged" />
<attribute name="date_reminded" />
<attribute name="date_due" />
<attribute name="date_response_due" />
<attribute name="declined" />
<attribute name="cancelled" />
<attribute name="automatic" />
<attribute name="quality" />

<attribute name="recommendation" />
<attribute name="competing_interest" />

</complexType>
</element>

<element name="form">
  <complexType>
    <sequence>
      <element ref="pkp:answer" minOccurs="0" maxOccurs="unbounded" />
    </sequence>
    <attribute name="title" />
  </complexType>
</element>

<element name="answer">
  <complexType>
    <attribute name="value" />
  </complexType>
</element>
```

```
<attribute name="viewable" />
</complexType>
</element>

<element name="queries">
  <complexType>
    <sequence>
      <element ref="pkp:query" minOccurs="0" maxOccurs="unbounded" />
    </sequence>
  </complexType>
</element>

<element name="query">
  <complexType>
    <sequence>
      <element ref="pkp:note" minOccurs="1" maxOccurs="unbounded" />
    </sequence>
  </complexType>
</element>

<element name="note">
  <complexType mixed="true">
    <attribute name="user" type="string" />
    <attribute name="date_created" />
    <attribute name="date_modified" />
    <attribute name="title" />
  </complexType>
</element>
</schema>
```

## XML con artículos

```
<?xml version="1.0"?>
<articles xmlns="http://pkp.sfu.ca" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://pkp.sfu.ca native.xsd">
  <article xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" date_submitted="2021-03-02"
status="3" submission_progress="0" current_publication_id="1" stage="production">
  <id type="internal" advice="ignore">1</id>
  <submission_file xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" stage="submission" id="1"
xsi:schemaLocation="http://pkp.sfu.ca native.xsd">
  <revision number="1" genre="Article Text" filename="Envio1Autor2.pdf" viewable="false"
date_uploaded="2021-03-04" date_modified="2021-03-04" filesize="12041" filetype="application/pdf"
uploader="autor2">
  <name locale="en_US">autor2, Envio1Autor2.pdf</name>
  <embed encoding="base64"></embed>
</revision>
</submission_file>
  <submission_file xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" stage="review_file" id="2"
xsi:schemaLocation="http://pkp.sfu.ca native.xsd">
  <revision number="1" source="1-1" genre="Article Text" filename="Envio1Autor2.pdf" viewable="false"
date_uploaded="2021-03-04" date_modified="2021-03-04" filesize="12041" filetype="application/pdf"
uploader="autor2">
  <name locale="en_US">Article Text, Envio1Autor2.pdf</name>
  <embed encoding="base64"></embed>
</revision>
</submission_file>
  <submission_file xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" stage="final" id="3"
xsi:schemaLocation="http://pkp.sfu.ca native.xsd">
  <revision number="1" genre="Article Text" filename="Envio1Autor2.pdf" viewable="false"
date_uploaded="2021-03-04" date_modified="2021-03-04" filesize="12041" filetype="application/pdf"
uploader="admin">
  <name locale="en_US">admin, Envio1Autor2.pdf</name>
  <embed encoding="base64"></embed>
</revision>
</submission_file>
  <submission_file xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" stage="copyedit" id="4"
xsi:schemaLocation="http://pkp.sfu.ca native.xsd">
  <revision number="1" source="3-1" genre="Article Text" filename="Envio1Autor2.pdf"
viewable="false" date_uploaded="2021-03-04" date_modified="2021-03-04" filesize="12041"
filetype="application/pdf" uploader="admin">
  <name locale="en_US">admin, Envio1Autor2.pdf</name>
  <embed encoding="base64"></embed>
</revision>
</submission_file>
  <submission_file xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" stage="production_ready"
id="5" xsi:schemaLocation="http://pkp.sfu.ca native.xsd">
  <revision number="1" source="4-1" genre="Article Text" filename="Envio1Autor2.pdf" viewable="false"
date_uploaded="2021-03-04" date_modified="2021-03-04" filesize="12041" filetype="application/pdf">
```

```
uploader="admin">
  <name locale="en_US">admin, Envio1Autor2.pdf</name>
  <embed encoding="base64"></embed>
</revision>
</submission_file>
<publication xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" locale="en_US" version="1"
status="3" primary_contact_id="1" url_path="" seq="0" date_published="2021-03-04" section_ref="ART"
access_status="0" xsi:schemaLocation="http://pkp.sfu.ca native.xsd">
  <id type="internal" advice="ignore">1</id>
  <title locale="en_US">Envio1Autor2</title>
  <abstract locale="en_US">&lt;p&gt;envio 1 autor 2&lt;/p&gt;</abstract>
  <copyrightHolder locale="en_US">Revista Destino</copyrightHolder>
  <copyrightYear>2021</copyrightYear>
  <authors xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://pkp.sfu.ca native.xsd">
    <author include_in_browse="true" user_group_ref="Author" seq="0" id="1">
      <givenname locale="en_US">Autor2</givenname>
      <email>autor2@gmail.com</email>
    </author>
  </authors>
  <issue_identification>
    <volume>1</volume>
    <number>number1</number>
    <year>2021</year>
    <title locale="en_US">issue1</title>
  </issue_identification>
</publication>
<participants>
  <participant mail="editorrevista1@gmail.com" user_group_ref="Journal editor"/>
  <participant mail="autor2@gmail.com" user_group_ref="Author"/>
</participants>
<stages>
  <stage id="1" name="submission">
    <queries/>
  </stage>
  <stage id="2" name="internalReview">
    <queries/>
  </stage>
  <stage id="3" name="externalReview">
    <queries/>
  </stage>
  <rounds>
    <round>
      <file id="2" number="1" stage="4" source="1-1" genre="Article Text" filename="Envio1Autor2.pdf"
viewable="false" date_uploaded="2021-03-04" date_modified="2021-03-04" filesize="12041"
filetype="application/pdf" uploader="autor2@gmail.com">
        <name locale="en_US">Article Text, Envio1Autor2.pdf</name>
      </file>
    </round>
  </rounds>
</stages>
```



```
</rounds>
</stage>
<stage id="4" name="copyEditing">
  <queries>
    <query>
      <note user="admin@gmail.com" date_created="2021-03-04 18:23:41" date_modified="2021-03-04
18:23:41" title="DiscussionCopyEdit">&lt;p&gt;message&lt;/p&gt;</note>
    </query>
  </queries>
</stage>
<stage id="5" name="production">
  <queries>
    <query>
      <note user="admin@gmail.com" date_created="2021-03-04 18:23:41" date_modified="2021-03-04
18:23:41" title="Production1discussion">&lt;p&gt;message&lt;/p&gt;</note>
    </query>
  </queries>
</stage>
</stages>
</article>
<article xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" date_submitted="2021-03-02"
status="3" submission_progress="0" current_publication_id="2" stage="production">
  <id type="internal" advice="ignore">2</id>
  <submission_file xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" stage="submission" id="6"
xsi:schemaLocation="http://pkp.sfu.ca native.xsd">
    <revision number="1" genre="Article Text" filename="Envio1Autor2.pdf" viewable="false"
date_uploaded="2021-03-04" date_modified="2021-03-04" filesize="12041" filetype="application/pdf"
uploader="autor2">
      <name locale="en_US">autor2, Envio1Autor2.pdf</name>
      <embed encoding="base64"></embed>
    </revision>
  </submission_file>
  <submission_file xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" stage="review_file" id="7"
xsi:schemaLocation="http://pkp.sfu.ca native.xsd">
    <revision number="1" source="6-1" genre="Article Text" filename="Envio1Autor2.pdf" viewable="false"
date_uploaded="2021-03-04" date_modified="2021-03-04" filesize="12041" filetype="application/pdf"
uploader="autor2">
      <name locale="en_US">Article Text, Envio1Autor2.pdf</name>
      <embed encoding="base64"></embed>
    </revision>
  </submission_file>
  <submission_file xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" stage="final" id="8"
xsi:schemaLocation="http://pkp.sfu.ca native.xsd">
    <revision number="1" genre="Article Text" filename="Envio1Autor2.pdf" viewable="false"
date_uploaded="2021-03-04" date_modified="2021-03-04" filesize="12041" filetype="application/pdf"
uploader="admin">
      <name locale="en_US">admin, Envio1Autor2.pdf</name>
      <embed encoding="base64"></embed>
    </revision>
  </submission_file>
</article>
```

```
</revision>
</submission_file>
<submission_file xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" stage="copyedit" id="9"
xsi:schemaLocation="http://pkp.sfu.ca native.xsd">
  <revision number="1" source="8-1" genre="Article Text" filename="Envio1Autor2.pdf" viewable="false"
date_uploaded="2021-03-04" date_modified="2021-03-04" filesize="12041" filetype="application/pdf"
uploader="admin">
  <name locale="en_US">admin, Envio1Autor2.pdf</name>
  <embed encoding="base64"></embed>
</revision>
</submission_file>
<submission_file xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" stage="production_ready"
id="10" xsi:schemaLocation="http://pkp.sfu.ca native.xsd">
  <revision number="1" source="9-1" genre="Article Text" filename="Envio1Autor2.pdf" viewable="false"
date_uploaded="2021-03-04" date_modified="2021-03-04" filesize="12041" filetype="application/pdf"
uploader="admin">
  <name locale="en_US">admin, Envio1Autor2.pdf</name>
  <embed encoding="base64"></embed>
</revision>
</submission_file>
<publication xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" locale="en_US" version="1"
status="3" primary_contact_id="2" url_path="" seq="0" date_published="2021-03-04" section_ref="ART"
access_status="0" xsi:schemaLocation="http://pkp.sfu.ca native.xsd">
  <id type="internal" advice="ignore">2</id>
  <title locale="en_US">Envio1Autor2</title>
  <abstract locale="en_US">&lt;p&gt;envio 1 autor 2&lt;/p&gt;</abstract>
  <copyrightHolder locale="en_US">Revista Destino</copyrightHolder>
  <copyrightYear>2021</copyrightYear>
  <authors xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://pkp.sfu.ca native.xsd">
    <author include_in_browse="true" user_group_ref="Author" seq="0" id="2">
      <givenname locale="en_US">Autor2</givenname>
      <email>autor2@gmail.com</email>
    </author>
  </authors>
  <issue_identification>
    <volume>1</volume>
    <number>number1</number>
    <year>2021</year>
    <title locale="en_US">issue1</title>
  </issue_identification>
</publication>
<participants>
  <participant mail="editorrevista1@gmail.com" user_group_ref="Journal editor"/>
  <participant mail="autor2@gmail.com" user_group_ref="Author"/>
</participants>
<stages>
  <stage id="1" name="submission">
```

```
<queries/>
</stage>
<stage id="2" name="internalReview">
  <queries/>
</stage>
<stage id="3" name="externalReview">
  <queries/>
  <rounds>
    <round>
      <file id="7" number="1" stage="4" source="6-1" genre="Article Text" filename="Envio1Autor2.pdf"
viewable="false" date_uploaded="2021-03-04" date_modified="2021-03-04" filesize="12041"
filetype="application/pdf" uploader="autor2@gmail.com">
        <name locale="en_US">Article Text, Envio1Autor2.pdf</name>
      </file>
    </round>
  </rounds>
</stage>
<stage id="4" name="copyEditing">
  <queries>
    <query>
      <note user="admin@gmail.com" date_created="2021-03-04 18:26:22" date_modified="2021-03-04
18:26:22" title="DiscussionCopyEdit">&lt;p&gt;message&lt;/p&gt;</note>
    </query>
  </queries>
</stage>
<stage id="5" name="production">
  <queries>
    <query>
      <note user="admin@gmail.com" date_created="2021-03-04 18:26:22" date_modified="2021-03-04
18:26:22" title="Production1discussion">&lt;p&gt;message&lt;/p&gt;</note>
    </query>
  </queries>
</stage>
</stages>
</article>
<article xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" date_submitted="2021-03-01"
status="4" submission_progress="0" current_publication_id="3" stage="submission">
  <id type="internal" advice="ignore">3</id>
  <submission_file xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" stage="submission" id="11"
xsi:schemaLocation="http://pkp.sfu.ca native.xsd">
    <revision number="1" genre="Article Text" filename="Envio1Autor1.pdf" viewable="false"
date_uploaded="2021-03-04" date_modified="2021-03-04" filesize="12041" filetype="application/pdf"
uploader="autor1">
      <name locale="en_US">autor1, Envio1Autor1.pdf</name>
      <embed encoding="base64"></embed>
    </revision>
  </submission_file>
  <publication xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" locale="en_US" version="1"
```

```
status="1" primary_contact_id="3" url_path="" seq="0" section_ref="ART" access_status="0"
xsi:schemaLocation="http://pkp.sfu.ca native.xsd">
  <id type="internal" advice="ignore">3</id>
  <title locale="en_US">Envio1</title>
  <abstract locale="en_US">&lt;p&gt;envio1 abstract&lt;/p&gt;</abstract>
  <authors xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://pkp.sfu.ca native.xsd">
    <author include_in_browse="true" user_group_ref="Author" seq="0" id="3">
      <givenname locale="en_US">Autor1</givenname>
      <email>autor1@gmail.com</email>
    </author>
  </authors>
</publication>
<participants>
  <participant mail="editorrevista1@gmail.com" user_group_ref="Journal editor"/>
  <participant mail="autor1@gmail.com" user_group_ref="Author"/>
</participants>
<stages>
  <stage id="1" name="submission">
    <queries/>
  </stage>
  <stage id="2" name="internalReview">
    <queries/>
  </stage>
  <stage id="3" name="externalReview">
    <queries/>
    <rounds/>
  </stage>
  <stage id="4" name="copyEditing">
    <queries/>
  </stage>
  <stage id="5" name="production">
    <queries/>
  </stage>
</stages>
</article>
</articles>
```

## Agradecimientos

A mi asesor Gonzalo, por haberme propuesto este proyecto, además de ser alguien que siempre estuvo para resolver mis dudas e inseguridades. Sin dudarlo, en mi opinión es un gran ejemplo a seguir tanto en lo profesional como en lo personal y espero que este trabajo pueda superar sus expectativas.

A Mauricio Nappa, Cecilia Rozemblum y Dolores García por haberme dado una gran ayuda con respecto a la redacción y por haberme instruido respecto al tema. Este proyecto no habría sido lo mismo sin la ayuda de ellos.

A mi directora Marisa, por sus consejos que aportaron a la realización de la tesis.

Y a todos mis amigos y mi familia que hicieron de gran sostén durante este proyecto y a lo largo de toda mi carrera.