https://doi.org/10.22201/dgtic.30618096e.2025.3.4.135 **Vol. 3, Núm. 4.** octubre-diciembre 2025, págs. 20 - 64

Adaptación de Open Journal Systems para el proceso de evaluación de recursos en la Red Universitaria de Aprendizaje

Mario Alberto Hernández Mayorga
Dirección General de Cómputo y de Tecnologías de Información y Comunicación
Universidad Nacional Autónoma de México
mariohm@unam.mx
ORCID: 0009-0002-5504-6009

Cristian Ricardo Ortega Ramírez
Dirección General de Cómputo y de Tecnologías de Información y Comunicación
Universidad Nacional Autónoma de México
cristian.ortega@comunidad.unam.mx
ORCID: 0009-0003-1073-6492

Rebeca Valenzuela Argüelles
Dirección General de Cómputo y de Tecnologías de Información y Comunicación
Universidad Nacional Autónoma de México
rebecav@unam.mx
ORCID: 0009-0002-8243-3258

Información del reporte:

Para citar este reporte técnico:

Hernández Mayorga, M. A., Ortega Ramírez, C.R. y Valenzuela Argüelles, R. (2025). Adaptación de Open Journal Systems para el proceso de evaluación de recursos en la Red Universitaria de Aprendizaje. *Cuadernos Técnicos Universitarios de la DGTIC, 3* (4) páginas (20 - 64). https://doi.org/10.22201/dgtic.30618096e.2025.3.4.135

Resumen

El proceso de evaluación para garantizar la calidad de los recursos educativos a publicar en la Red Universitaria de Aprendizaje ha sido una prioridad de este proyecto desde su planteamiento. Ante el crecimiento del equipo evaluador y el volumen de propuestas, se analizaron diferentes plataformas digitales con el propósito de mejorar dicho proceso, lo cual motivó la búsqueda de una solución que permitiera sistematizar sus distintas etapas, que se habían realizado de manera manual

https://doi.org/10.22201/dgtic.30618096e.2025.3.4.135 **Vol. 3,Núm. 4. octubre-diciembre** 2025, págs. 21 - 64

durante varios años. La solución consistió en personalizar un sistema de software libre, originalmente diseñado para la publicación de revistas académicas, mediante el desarrollo de *plugins* que permitieron incorporar información estructurada conforme a los requerimientos del proyecto. Se documentaron los desafíos técnicos del proceso, se construyó un entorno de desarrollo controlado y se llevaron a cabo pruebas iterativas con usuarios para ajustar el flujo de trabajo y verificar el funcionamiento de los *plugins* desarrollados. La integración de funcionalidades específicas permitió registrar, evaluar y dictaminar recursos educativos con mayor eficiencia, asegurando al mismo tiempo la trazabilidad del proceso. El sistema adaptado se utilizó en una convocatoria institucional y demostró su viabilidad en un contexto operativo real, aportando una mejora sustancial frente a los métodos previos. Esta experiencia mostró cómo es posible adecuar herramientas tecnológicas existentes a necesidades particulares sin comprometer su estabilidad ni sus principios de diseño.

Palabras clave:

Evaluación de recursos educativos, desarrollo de *plugins*, *plugins* para OJS, adaptación de software libre.

Abstract

The evaluation process to ensure the quality of the educational resources to be published on the Red Universitaria de Aprendizaje has been a priority of this project since its establishment. In response to the growth of the evaluation team and the volume of proposals, different digital platforms were analyzed with the aim of improving the process. This motivated the search for a solution that would allow systematizing their different stages, which had been done manually for several years. The solution consisted in personalizing an open-source system, originally designed for academic magazine publication, through the development of plugins that allowed to incorporate structured information pursuant to the project requirements. The technical challenges of the process were documented, a controlled development environment was built and iterative testing was done with users to adjust the workflow and verify the operation of developed plugins. Specific functionalities integration allowed to register, evaluate and rule educational resources with greater efficiency, ensuring at the same time the traceability of the project. The adapted system was used in an institutional call and demonstrated its viability in a real operative context, contributing a substantial improvement in comparison of previous methods. This experience showed how it is possible to adequate existing technological tools for particular needs without compromising its stability nor its design principles.

Keywords:

Educational resources evaluation, plugin development, plugins for OJS, open source software adaptation.

1. INTRODUCCIÓN

La Red Universitaria de Aprendizaje (RUA) es un proyecto que nació en el 2012 con la intención de reunir y publicar recursos digitales asociados a los planes de estudio de la Universidad Nacional Autónoma de México (UNAM).

La RUA es una plataforma tecnológica que permite apoyar el proceso de enseñanza y aprendizaje en diferentes modalidades educativas, por medio de materiales educativos digitales, libres y gratuitos, con

https://doi.org/10.22201/dgtic.30618096e.2025.3.4.135 **Vol. 3,Núm. 4. octubre-diciembre** 2025, págs. 22 - 64

la intención de que el docente proporcione el contexto a cada uno de ellos dentro de la dinámica prevista para su clase.

Este sistema es un importante escaparate para la producción de los docentes de la institución, ya que brinda la oportunidad de publicar, en un mismo espacio, todo el trabajo que se genera al respecto, de modo que sea aprovechado por la comunidad universitaria y por todas aquellas personas que estén interesadas en los diversos temas abordados.

A diferencia de otros sistemas de su categoría, los cuales clasifican los recursos educativos de forma temática, en la RUA, se planteó que los materiales apoyaran a las asignaturas que se imparten en la UNAM desde la concepción de este proyecto, por lo que éstos se clasifican de acuerdo con los programas correspondientes con el objetivo de que su contenido, nivel, orientación y calidad sean congruentes con lo que se imparte en cada uno de ellos; esto es posible por medio de un proceso de evaluación.

La gestión para el envío, evaluación y dictamen de recursos que se publican en la RUA se había llevado a cabo de manera manual hasta ahora, utilizando hojas de cálculo para el registro de información y el correo electrónico como medio principal de comunicación entre autores y evaluadores. Este proceso se había afinado con el tiempo hasta tenerlo cuidadosamente sistematizado, lo cual había permitido avanzar en la organización y seguimiento de los recursos aprobados, rechazados o condicionados; sin embargo, implicaba retos en términos de tiempo, esfuerzo y capacitación necesarios para asegurar un flujo de trabajo controlado, ordenado y estandarizado a lo largo de las distintas fases del mismo.

El objetivo de este documento es presentar el proceso de análisis, diseño, desarrollo e implementación que se llevó a cabo, con la finalidad de automatizar el proceso de recepción, evaluación y comunicación de resultados para las propuestas de publicación de recursos educativos en la Red Universitaria de Aprendizaje.

2. HISTORIA DEL PROCESO DE EVALUACIÓN EN LA RUA

La Red Universitaria de Aprendizaje es un proyecto institucional que ha tenido que evolucionar en diversos aspectos para poder atender las necesidades que se han ido presentando a lo largo de su historia dentro de la UNAM, con el fin de mantener su prestigio como uno de los principales espacios digitales que ofrecen recursos educativos de calidad para la comunidad universitaria y la sociedad en general.

Al inicio, la recopilación de recursos se realizó por un grupo de curadores y catalogadores[1] que se dedicaron a analizar los planes y programas de estudio de la UNAM, quienes tenían por objetivo identificar recursos adecuados. En ese momento, la prioridad fue cubrir con al menos un recurso por tema en cada uno de los programas del sistema de bachillerato de la UNAM para que éste cumpliera con la lista de cotejo que se definió con el fin de asegurar la calidad inicial mínima necesaria (Valenzuela, en prensa).

1 En ese momento, las actividades de curación y catalogación se realizaron por un solo equipo de trabajo capacitado para ambas tareas. El equipo estaba integrado por un bibliotecólogo y sus demás miembros eran de diferentes disciplinas, como Matemáticas, Pedagogía, Historia, Física, entre otras. La curación, como se indica en el cuerpo del texto, se realizó con base en los temas indicados en los planes y programas de estudio correspondientes. Estos recursos se catalogaron por dicho equipo y, más tarde, docentes de las asignaturas evaluaron la pertinencia de cada uno de los recursos propuestos para definir si debían estar publicados o no.

https://doi.org/10.22201/dgtic.30618096e.2025.3.4.135 **Vol. 3,Núm. 4. octubre-diciembre** 2025, págs. 23 - 64

Más tarde, los recursos recuperados por los catalogadores eran evaluados en campañas con docentes de las respectivas asignaturas y eran ellos quienes definían los materiales que cumplían con lo necesario para su publicación. Es así que, de 2015 a 2017, se llevaron a cabo siete campañas de evaluación en las que se analizaron casi diez mil recursos por cerca de quinientos profesores.

A partir de 2016, el énfasis se hizo en la recuperación de recursos que se habían producido por académicos de la UNAM en los diferentes proyectos apoyados por la misma institución. Como resultado de dichos proyectos, se cuenta con una alta producción de materiales en diferentes medios que, en varios casos, no tenían un espacio específico en el cual publicarse. Hoy en día, la RUA es uno de los principales sitios para ello.

En 2017, se comenzaron a recibir los proyectos con un nuevo esquema y se implementó el siguiente proceso para su recepción y evaluación:

1. Recepción y análisis de documentación de la propuesta.
2. Revisión técnica.
3. Evaluación académica en comité de pares.
4. Evaluación de la calidad del servicio brindado por el equipo de la RUA durante el proceso de publicación.
5. Publicación de recursos avalados.

En 2019, se desarrolló la documentación oficial para normar dicho proceso, con el fin de dar claridad y transparencia a los docentes en un espacio específico de la propia RUA: la sección *Publica recursos*.

A partir de 2023, la propuesta de materiales se realizó mediante una convocatoria anual con un proceso de evaluación diferente, lo cual permitió tener un periodo determinado de recopilación, análisis, evaluación y publicación, además de agilizar la capacidad de respuesta a los autores.

Sin embargo, con este nuevo proceso, la cantidad de participantes en la evaluación creció y era notorio que, aunque las hojas de cálculo eran útiles para llevar el control y seguimiento de las actividades, su uso requería de una organización rigurosa y, en algunas ocasiones, dificultaba obtener una visión clara del avance en el proceso de evaluación y publicación de los recursos educativos, debido a la cantidad de información registrada sobre cada recurso (metadatos, datos de contacto del responsable del recurso, dictámenes, entre otros).

3. METADATOS DE LA RUA

Los metadatos, de acuerdo con Miller (2022), pueden ser definidos como información que describe otros recursos informativos tales como libros, archivos de audio, conjuntos de datos científicos, imágenes digitales, etcétera.

La RUA emplea una adaptación del estándar de metadatos Learning Object Metadata (LOM), el cual fue desarrollado específicamente para describir objetos de aprendizaje, incluyendo metadatos pedagógicos esenciales como el nivel educativo y el contexto de uso, entre otros.

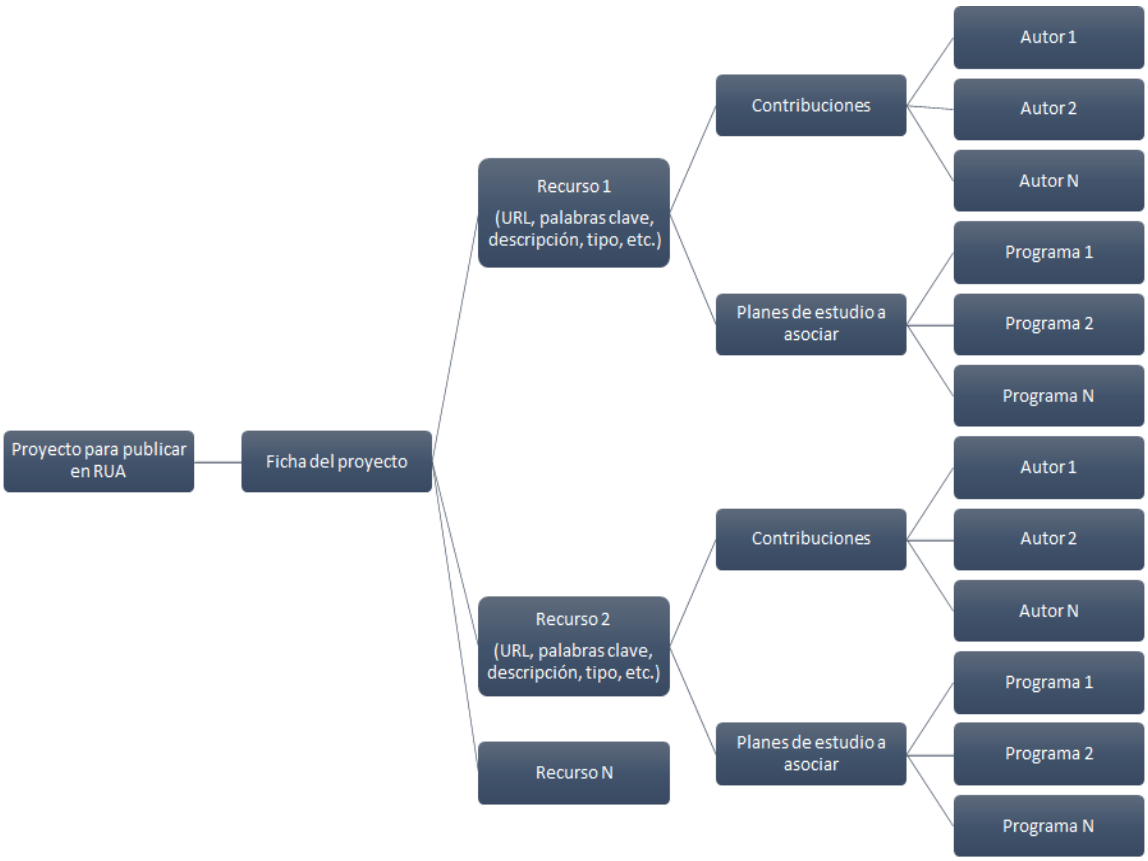
https://doi.org/10.22201/dgtic.30618096e.2025.3.4.135 **Vol. 3,Núm. 4. octubre-diciembre** 2025, págs. 24 - 64

La adaptación de LOM para la RUA contempla un conjunto aproximado de 28 metadatos distribuidos en las siguientes categorías: General, Ciclo de vida, Técnica, Uso educativo, Derechos, Control, Clasificación y Portada. Más adelante, se profundizará en el hecho de que estos elementos son fundamentales para garantizar una adecuada clasificación, búsqueda y reutilización de los recursos.

Para el registro de los trabajos, se solicitó a los autores únicamente ciertos metadatos necesarios para la catalogación básica en la RUA, considerando que, para ellos, podía resultar complicado el llenado de todos aquellos campos obligatorios en la catalogación completa.

Figura 1

Registro de propuestas por recurso



Se muestran en la Figura 1, de manera general, los metadatos requeridos para el registro de propuestas por recurso y se presentan de manera detallada en el Anexo A.

https://doi.org/10.22201/dgtic.30618096e.2025.3.4.135 **Vol. 3,Núm. 4. octubre-diciembre** 2025, págs. 25 - 64

4. ANÁLISIS DE HERRAMIENTAS

Para solventar el inconveniente mencionado sobre la manera de gestionar la evaluación de las propuestas para la publicación de recursos educativos, se planteó la posibilidad de sistematizar el proceso por medio de alguna plataforma que permitiera brindar el seguimiento necesario.

4.1 ANÁLISIS DE OJS

La primera opción fue utilizar un sistema gestor de artículos cuyo flujo editorial resultaba cercano al proceso que se llevaba a cabo en la RUA. En ese sentido, se identificó al Open Journal Systems (OJS) como uno de los principales softwares de gestión y publicación de revistas académicas, el cual fue desarrollado por el Public Knowledge Project y lanzado en 2002 (Public Knowledge Project, s.f.-a).

La plataforma OJS ha sido diseñada específicamente para reducir el tiempo dedicado a las tareas de gestión asociadas a la edición de una revista, al tiempo que mejora el mantenimiento de registros y la eficiencia de los procesos editoriales (Willinsky, 2005). Además, este tipo de infraestructura digital acompaña todos y cada uno de los pasos del editor, apoyando de manera integral las tareas editoriales en un ritmo y magnitud que no podrían lograrse sin su apoyo (Hartstein & Blümel, 2021).

De acuerdo con Tabatadze (2024), dentro los componentes clave de OJS destacan:

- Gestión integral del flujo editorial: permite gestionar todo el proceso editorial desde la recepción de manuscritos hasta la publicación final, facilitando la comunicación entre los distintos actores que intervienen en el proceso.
- Automatización de procesos: la plataforma agiliza tareas clave como la asignación de revisores, el control de revisiones y la generación de metadatos, reduciendo la carga de trabajo manual.
- Accesibilidad y personalización: al ser un software de código abierto, es accesible para cualquier institución y puede adaptarse a las necesidades específicas de éstas, tanto en diseño como en funcionalidad mediante el uso de *plugins*.

Durante el análisis de la plataforma OJS como posible herramienta para la gestión de propuestas de recursos educativos para la RUA, se identificó una limitación clave relacionada con los metadatos. OJS utiliza el estándar de metadatos Dublin Core, ampliamente reconocido en la gestión de información académica y bibliográfica, el cual permite describir los artículos publicados mediante 15 elementos clave como título, autor, resumen y palabras clave, entre otros. Aunque OJS cubre la mayoría de los requerimientos del proceso editorial de la RUA, no ofrece de forma nativa una manera sencilla y directa de agregar campos de metadatos personalizados sin recurrir a modificaciones en el código o al desarrollo de *plugins*. Esta limitación se vuelve especialmente relevante en el contexto de la RUA, donde se requería adaptar un esquema específico de metadatos basado en LOM para una descripción pedagógica más precisa de los recursos. Además, como se advierte en los foros oficiales de PKP, modificar la estructura de metadatos sin un enfoque cuidadosamente diseñado puede generar inconsistencias y afectar la estabilidad del sistema (Asmecher, 2017).

Se decidió explorar herramientas alternativas que pudieran ofrecer mayor flexibilidad al considerar las limitaciones descritas anteriormente en la personalización de metadatos en OJS. En este contexto, se realizó el análisis de la plataforma Kotahi.

https://doi.org/10.22201/dgtic.30618096e.2025.3.4.135 **Vol. 3,Núm. 4. octubre-diciembre** 2025, págs. 26 - 64

4.2 ANÁLISIS DE KOTAHI

Kotahi es una plataforma adaptable y personalizable, diseñada para investigadores, que satisface las diversas necesidades de Publicación-Revisión-Curación (PRC) y los modelos emergentes de publicación académica. La flexibilidad de Kotahi permite ajustar el sistema para satisfacer necesidades específicas (Kotahi, s.f.). Esta descripción de la plataforma permitió considerar que se trataba de una alternativa a OJS, dada su flexibilidad para ser adaptada a diversas aplicaciones académicas.

Sin embargo, cuando se llevó a cabo la instalación para realizar las pruebas de la misma, se identificó que Kotahi está relacionada fuertemente con el sistema ORCID (Open Researcher and Contributor ID, s.f.), en el cual se requiere un identificador para cada autor. Esto no resultaba práctico para el caso de la RUA, en donde quienes proponen trabajos son principalmente docentes y la gran mayoría de ellos no cuentan con dicho dato. Además, para el caso específico de la RUA, el contar con un ORCID no es algo esencial, dado que no se realiza la publicación de un trabajo de investigación, sino de un recurso digital con fines didácticos, lo cual no precisa los mismos estándares que una publicación científica.

Con estos resultados, se decidió retomar y profundizar en la posibilidad técnica de adaptar OJS para incorporar los metadatos que no contempla de forma nativa y que son requeridos para el registro de recursos en la RUA. El análisis comparativo para la identificación de los metadatos a incorporar se encuentra en el Anexo A.

5. DESARROLLO TÉCNICO

La documentación oficial de OJS recomienda el uso de *plugins* como mejor práctica para ampliar las capacidades de la plataforma. Este enfoque permite añadir funcionalidades adicionales sin modificar el núcleo del sistema, lo que facilita las actualizaciones a nuevas versiones de OJS sin riesgo de perder personalizaciones ni introducir errores en el sistema base. Además, los *plugins* favorecen una arquitectura modular y flexible, permitiendo la activación, desactivación y actualización independiente de las extensiones, lo cual asegura que el funcionamiento general de la plataforma no se vea afectado (Public Knowledge Project, 2025).

5.1 METODOLOGÍA

En este contexto, el uso de *pair programming* (programación en pareja) resulta altamente relevante. Esta metodología ha demostrado ser particularmente eficaz en equipos que enfrentan tareas con un alto nivel de novedad. Según estudios recientes, la *pair programming* fomenta que los miembros del equipo compartan una comprensión común del proyecto y se apoyen mutuamente, lo que contribuye significativamente al rendimiento del equipo frente a retos complejos y desconocidos (Kude et al., 2019). Al trabajar de manera simultánea en el mismo código, los desarrolladores no sólo resuelven problemas de forma más rápida, sino que también comparten conocimientos y refuerzan la calidad del código mediante interacciones continuas.

La colaboración estrecha que ofrece la *pair programming* permite una mejor integración de nuevas funcionalidades como los *plugins*, ya que la constante retroalimentación y la solución conjunta de problemas aceleran el proceso de desarrollo. Además, esta técnica ayuda a mejorar la eficiencia del equipo al enfrentar la complejidad de las tareas, favoreciendo la innovación colaborativa y el intercambio

https://doi.org/10.22201/dgtic.30618096e.2025.3.4.135 **Vol. 3,Núm. 4. octubre-diciembre** 2025, págs. 27 - 64

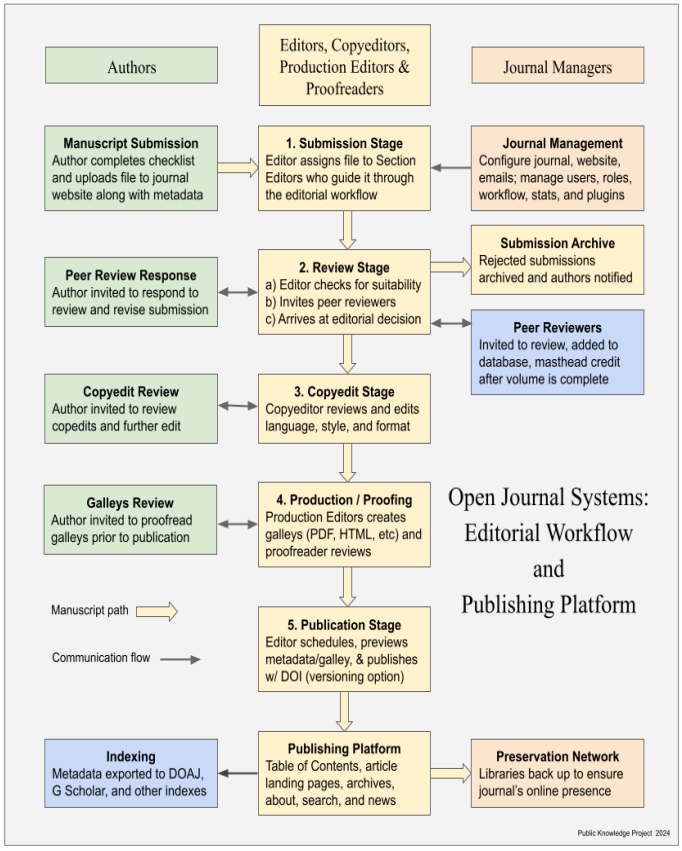
de conocimientos. Estudios previos han mostrado que este enfoque aumenta la productividad, la calidad del código y promueve la innovación en el desarrollo de software (Tan et al., 2024). Por ello, en este proyecto, la *pair programming* se presenta como una metodología adecuada para el desarrollo de los *plugins* necesarios que amplíen las capacidades de OJS sin comprometer la estabilidad de la plataforma.

5.1.1 ANÁLISIS

En esta fase, se llevó a cabo una revisión de la documentación oficial de OJS, con el objetivo de comprender el flujo de trabajo que utiliza la plataforma, prestando especial atención a la forma en que se gestionan los metadatos a lo largo del proceso editorial. Esta revisión permitió identificar en qué momentos se capturan, editan y exponen los metadatos dentro del sistema, información clave para determinar los puntos donde sería necesaria una adaptación (ver Figura 2).

Figura 2

Flujo de trabajo de OJS



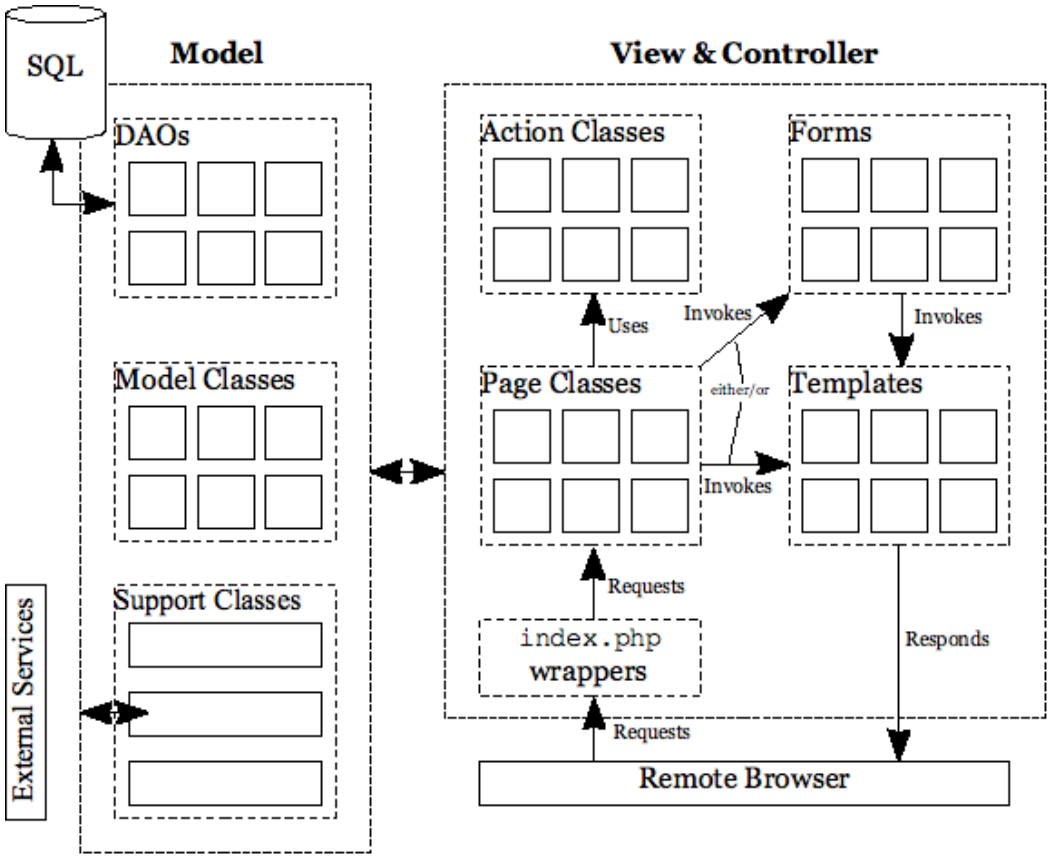
Nota: De Open Journal Systems, por Public Knowledge Project, s.f.-c (https://pkp.sfu.ca/wp-content/uploads/2024/06/OJS-Workflow-and-Platform-2024.png)

https://doi.org/10.22201/dgtic.30618096e.2025.3.4.135 **Vol. 3,Núm. 4. octubre-diciembre** 2025, págs. 28 - 64

Paralelamente, se realizó un análisis técnico de la arquitectura y los componentes principales de OJS, con el fin de entender cómo se estructura internamente la plataforma y bajo qué principios opera su sistema de extensiones. Este conocimiento fue fundamental para diseñar e implementar los *plugins* requeridos de manera compatible con el núcleo de OJS (ver Figura 3).

Figura 3

Componentes y arquitectura de OJS



Nota: De Open Journal Systems: Technical documentation. Introduction, de Public Knowledge Project, s.f.-d (https://docs.pkp.sfu.ca/ojs-2-technical-reference/en/2_introduction)

Además, OJS define diferentes categorías de *plugins* que permiten extender la funcionalidad de la plataforma de manera controlada. Estas categorías determinan en qué momento del ciclo de vida del sistema se cargan y qué aspectos del comportamiento o de la interfaz pueden modificar. Entre las principales categorías, se encuentran los *plugins* de bloques, de informes, de metadatos, de autenticación, de temas, de publicación, entre otros.

Dada la anterior información y a partir del análisis de los requerimientos específicos para la gestión de metadatos adicionales en los recursos educativos propuestos para la RUA, se identificó que sería necesario desarrollar cuatro *plugins* personalizados (ver Tabla 1). Cada uno de ellos responde a una necesidad particular del proceso de captura, almacenamiento, visualización o exposición de metadatos, y su diseño

https://doi.org/10.22201/dgtic.30618096e.2025.3.4.135 **Vol. 3,Núm. 4. octubre-diciembre** 2025, págs. 29 - 64

se alineó con las categorías definidas por OJS para asegurar su correcta integración y compatibilidad con la plataforma. Esta estrategia permitió mantener una arquitectura flexible, escalable y conforme a las buenas prácticas recomendadas por la documentación oficial del sistema.

Tabla 1

Definición de plugins a desarrollar

Nombre	Descripción	Categoría de plugin
ruaMetadata	Incluye los metadatos personalizados en los formularios de nuevo envio[2] y edición de envío	Genérico
ruaTheme	Sobreescribe las plantillas para mostrar los metadatos en un concreto y agrupados por categoría	Tema
ruaReport	Exporta los datos de los envíos incluyendo los metadatos personalizados	Reporte
ruaOAIDim	Expone los metadatos personalizados en el formato DIM mediante el protocolo OAI-PMH	oaiMetadataFormats

5.1.2 IMPLEMENTACIÓN

Para el desarrollo y prueba de los *plugins*, se optó por crear un entorno de trabajo basado en contenedores. Esta elección respondió a la necesidad de disponer de un entorno controlado y reproducible a lo largo de las distintas etapas del proyecto.

Se utilizó la tecnología de contenedores Docker, ya que, gracias a su naturaleza de código abierto y a la gestión centralizada a través de registros de imágenes, facilita la creación de entornos uniformes para desarrollo, pruebas y producción (Wang, 2022). Una vez configurado el entorno de desarrollo, este puede ser encapsulado como una imagen de contenedor [3], almacenado en un repositorio y posteriormente desplegado en otros entornos sin modificaciones, lo que garantiza la consistencia funcional y reduce significativamente los errores derivados de diferencias en la configuración del sistema.

Este entorno controlado se construyó a partir del proyecto oficial pkp/docker-ojs[4], alojado en GitHub, cuya arquitectura base sólo contempla dos servicios. Sin embargo, para este proyecto, se implementaron tres servicios interconectados mediante una red interna. La elección de las imágenes de contenedor fue crucial para asegurar la compatibilidad: el servicio ojs utilizó la imagen pkpofficial/ojs con la etiqueta

2 Un envío en OJS es el registro formal de un trabajo académico presentado por un autor a través del sistema, el cual incluye la carga del manuscrito y los metadatos correspondientes.

3 Una imagen de contenedor es un archivo ejecutable e independiente que se utiliza para crear un contenedor. Esta imagen de contenedor contiene las bibliotecas, las dependencias y los archivos que el contenedor necesita para ejecutarse.

4 Proyecto base para el entorno de desarrollo (https://github.com/pkp/docker-ojs), actualmente archivado. Los esfuerzos de provisión de imágenes Docker para herramientas PKP se consolidaron en el nuevo repositorio (https://github.com/pkp/containers).

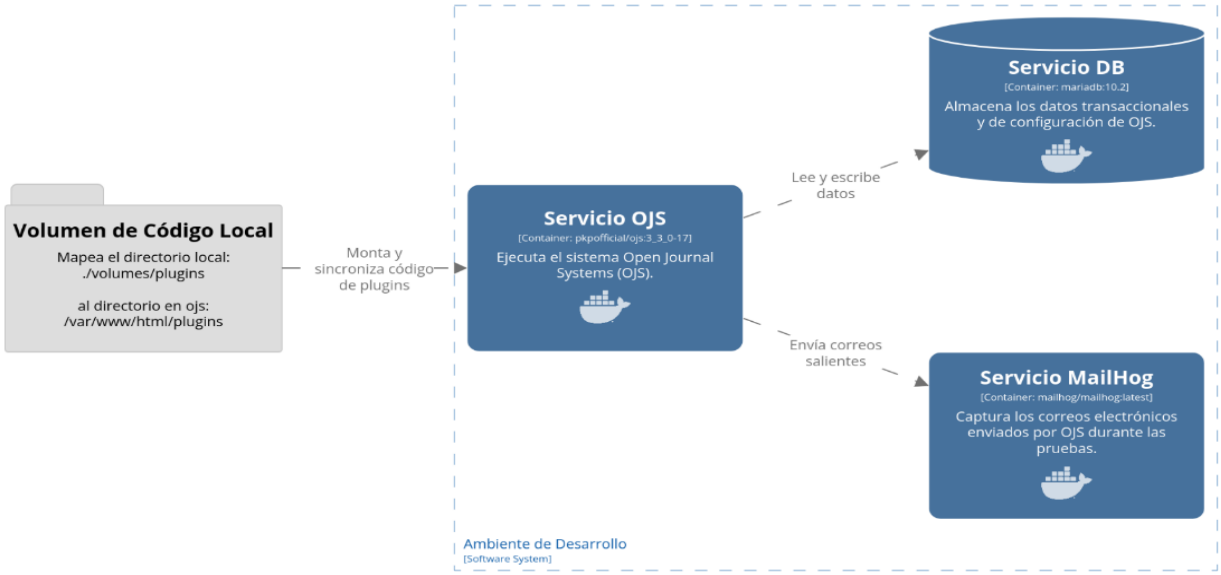
https://doi.org/10.22201/dgtic.30618096e.2025.3.4.135 **Vol. 3,Núm. 4. octubre-diciembre** 2025, págs. 30 - 64

3_3_0-17; el servicio db utilizó la imagen mariadb:10.2; y, de forma adicional, se incorporó el servicio mailhog con la imagen mailhog/mailhog:latest, esencial para capturar los correos electrónicos enviados por OJS durante las pruebas.

La estrategia clave para la programación fue el mapeo de volúmenes: se implementó un *volume mount* que replicó el directorio interno de *plugins* del contenedor (/var/www/html/plugins) en el directorio local del equipo anfitrión (./volumes/plugins). Esta disposición permitió editar el código fuente directamente sobre el sistema de archivos local, lo que integró de forma fluida el versionado del código fuente de los *plugins* con el flujo de trabajo de desarrollo y aseguró la trazabilidad de las modificaciones. En la Figura 4, se muestra un esquema del entorno de desarrollo.

Figura 4

Entorno de desarrollo



Al contar con un entorno controlado y replicable, fue posible centrarse en la implementación funcional de los *plugins*, asegurando que su comportamiento fuera consistente a lo largo de las distintas fases de prueba. A continuación, se detallan los aspectos clave del desarrollo de los *plugins*, incluyendo su estructura, integración con la arquitectura de OJS y las herramientas utilizadas durante el proceso.

5.1.3 DESARROLLO DE PLUGINS

Para este proyecto, se aplicó la técnica *Driver-Navigator* de *Pair Programming* descrita por Bolboacă (2021), ajustada a la dinámica particular del equipo de desarrollo. La estrategia consistió en asignar a cada desarrollador la responsabilidad principal sobre dos de los cuatro *plugins*, manteniendo dicha responsabilidad independientemente del rol (*driver* o *navigator*) que desempeñaba durante las sesiones de desarrollo colaborativo. De este modo, cada desarrollador tuvo la tarea de garantizar la correcta funcionalidad e integración de los componentes desarrollados en los *plugins* de los que era responsable.

https://doi.org/10.22201/dgtic.30618096e.2025.3.4.135 **Vol. 3,Núm. 4. octubre-diciembre** 2025, págs. 31 - 64

Durante las sesiones de trabajo conjunto, los roles de *driver* y *navigator* se alternaron a lo largo del desarrollo de todos los *plugins*. El *driver* se enfocó en ejecutar las implementaciones acordadas, aplicar correcciones y validar los resultados en tiempo real; mientras que el *navigator* revisó el código conforme se escribía, detectó posibles problemas de integración y propuso soluciones alternativas. Esta dinámica fortaleció la colaboración en un entorno donde el equipo tenía experiencia limitada con OJS.

***Plugin* ruaMetadata**

Este *plugin* pertenece a la categoría de *plugins* genéricos. Este tipo de *plugins* son extensiones diseñadas para añadir funcionalidades personalizadas que no encajan en otras categorías de *plugins* como temas, bloques o reportes. Permiten engancharse a distintos puntos de extensión del sistema (*hooks*), para modificar comportamientos, insertar *scripts*, validar formularios, interactuar con servicios externos o automatizar tareas específicas del flujo editorial. A diferencia de otros tipos de *plugins* más estructurados, los genéricos ofrecen libertad para implementar lógica personalizada. Esta versatilidad los convierte en una herramienta poderosa para adaptar OJS a necesidades particulares de una revista o institución.

Este *plugin* se desarrolló con el objetivo de satisfacer la necesidad de agregar los metadatos del estándar LOM adaptado para la RUA que no están contemplados dentro del esquema Dublin Core utilizado por OJS.

La primera etapa del desarrollo de este *plugin* consistió en identificar los puntos de extensión (*hooks*) adecuados para insertar los campos adicionales en el formulario correspondiente. Una vez identificados, se implementó la lógica necesaria para incorporar los nuevos metadatos y asegurar su almacenamiento persistente en la base de datos.

Un reto significativo en este proceso fue la identificación de los *hooks* y formularios adecuados, ya que muchos de ellos no están documentados oficialmente, lo que requirió una exploración detallada del código fuente de OJS. Aunque la guía de desarrollo de *plugins* proporciona algunos ejemplos y menciona los *hooks* más comunes, la lista no es exhaustiva. De hecho, la propia documentación de OJS recomienda buscar directamente en el código fuente la llamada a los *hooks* para descubrir todos los puntos de extensión disponibles en la versión 3.3. Esta limitación documental obligó al equipo de desarrollo a realizar un análisis manual del núcleo del sistema, rastreando el flujo de ejecución de las interfaces de usuario y los formularios involucrados en la edición de metadatos. Dicha tarea implicó no sólo localizar los *hooks* adecuados (ver Tabla 2), sino también comprender su contexto de ejecución y la estructura interna del sistema para insertar correctamente la lógica personalizada del *plugin* sin afectar la estabilidad general de la plataforma.

https://doi.org/10.22201/dgtic.30618096e.2025.3.4.135 **Vol. 3,Núm. 4. octubre-diciembre** 2025, págs. 32 - 64

Tabla 2

Hooks implementados en el plugin ruaMetadata

Nombre del Hook	Descripción	Rol del sistema que ocupa el hook
Schema::get::submission	Permite agregar nuevos campos personalizados al esquema del envío.	Autor
Templates::Submission::SubmissionMetadataForm::AdditionalMetadata	Se usa para inyectar contenido personalizado en la plantilla del formulario de metadatos del envío en el *backend* de OJS. Está pensado para complementar campos personalizados agregados desde el esquema con *hooks* como Schema::get::submission	Autor
submissiondao::getAdditionalFieldNames	Permite a los plugins registrar campos adicionales personalizados para que sean incluidos automáticamente en las operaciones de carga y almacenamiento del envío. Es necesario cuando se agregan campos personalizados al esquema de un envío y se requiere que esos campos persistan correctamente en la base de datos.	Autor
submissionsubmitstep3form::execute	Permite intervenir en el momento en que el sistema guarda los metadatos del envío, para procesar o guardar campos personalizados que el autor haya llenado.	Autor y Editor
submissionsubmitstep3form::initdata	Se dispara cuando el sistema inicializa los datos del formulario del paso 3 del proceso de registro. Su propósito es precargar en el formulario los datos existentes del envío. Es necesario cuando se agregan campos personalizados y se requiere mostrar la información capturada previamente en el formulario.	Autor y Editor
submissionsubmitstep3form::readuservars	Permite que campos personalizados agregados al formulario de registro sean leídos correctamente desde la solicitud del usuario, y estén disponibles para ser procesados y guardados.	Autor y Editor

https://doi.org/10.22201/dgtic.30618096e.2025.3.4.135 Vol. 3,Núm. 4. octubre-diciembre 2025, págs. 33 - 64

Nombre del Hook	Descripción	Rol del sistema que ocupa el hook
Form::config::before	Permite modificar la configuración de cualquier formulario basado en componentes antes de que se muestre en pantalla, lo que posibilita agregar o modificar campos sin alterar directamente el código del sistema.	Editor
Publication::edit	Permite intervenir justo antes de que se guarde una publicación en OJS, lo que es útil para aplicar validaciones, establecer valores por defecto o modificar datos personalizados definidos en el esquema.	Editor
Templates::View::RuaMetadata	Hook de elaboración propia que permite modificar la plantilla que muestra los metadatos de un envío a los revisores para incluir los metadatos adicionales agregados al esquema.	Revisor
Common::UserDetails:: AdditionalItems	Permite extender la información mostrada en las vistas de detalles de usuarios en el backend de OJS. Es utilizado para mostrar campos personalizados o datos derivados del comportamiento del usuario.	Autor
Templates::View:: RegistrationForm	Permite inyectar contenido personalizado en el formulario de registro de usuario de OJS. Es útil para agregar campos adicionales o mostrar contenido contextual.	—
Templates::View:: ReviewerSearch	Permite agregar contenido personalizado a la interfaz de búsqueda de revisores, útil para mostrar instrucciones, advertencias o complementos visuales.	Editor

En el Anexo B, se ilustra la diferencia entre los formularios originales para el registro y edición de un envío, los formularios modificados que incluyen los campos para la captura de los metadatos adicionales y fragmentos relevantes del código implementado.

Plugin ruaTheme

Este *plugin* pertenece a la categoría de Tema. Estos *plugins* son extensiones diseñadas específicamente para modificar la apariencia visual de una revista, sin alterar su lógica de funcionamiento. Esto favorece

https://doi.org/10.22201/dgtic.30618096e.2025.3.4.135 **Vol. 3,Núm. 4. octubre-diciembre** 2025, págs. 34 - 64

una arquitectura limpia, donde las personalizaciones visuales se mantienen separadas del núcleo de OJS, facilitando el mantenimiento y la actualización del sistema.

Dichos *plugins* permiten cambiar colores, tipografías, distribución de bloques, plantillas y estilos para adaptar el diseño del sitio a una identidad visual particular. Los temas pueden heredar de otros temas base y sobrescribir estilos o plantillas específicas.

En este caso particular, el *plugin* fue desarrollado con el objetivo de organizar visualmente la información en los formularios de metadatos. Específicamente, se requería agrupar los campos en secciones temáticas: recurso, proyecto, plan de estudio y notas. Para lograr esta organización, fue necesario sobrescribir secciones específicas de las plantillas existentes de la plataforma y, en ciertos casos, reemplazar completamente las plantillas originales con versiones adaptadas desde el *plugin*.

El principal desafío en este proceso consistió en modificar la interfaz administrativa de OJS, ya que los *plugins* de temas están diseñados principalmente para alterar la apariencia de la interfaz pública y no para intervenir en la administrativa. A partir de la versión 3 de OJS, este tipo de personalización presenta mayores dificultades y riesgos técnicos. La documentación oficial advierte que realizar cambios en esta sección del sistema conlleva un alto riesgo de comprometer su estabilidad, por lo que únicamente se recomienda a desarrolladores con experiencia avanzada y un conocimiento profundo de la arquitectura interna de la plataforma (Public Knowledge Project, s. f.-b).

Aunque es posible introducir modificaciones visuales en la interfaz administrativa, como sustituir algunas plantillas o aplicar estilos propios, esta capacidad no está bien documentada y cualquier intervención puede tener efectos colaterales difíciles de prever o rastrear. Para reducir los riesgos asociados a esta tarea, se llevó a cabo un análisis minucioso del código fuente de los componentes relevantes con el fin de identificar con precisión los elementos que podían adaptarse sin afectar el funcionamiento general del sistema. Esto permitió garantizar tanto la estabilidad de la plataforma como la implementación exitosa de una apariencia coherente en la interfaz administrativa.

En la Tabla 3, se listan las plantillas adaptadas y la modificación principal.

Tabla 3

Plantillas y modificación principal

Plantilla	Descripción	Modificación
submission/form/step3.tpl	Esta plantilla corresponde al formulario del paso 3 del registro de un envío, el cual está relacionado con la captura de sus metadatos.	Se modificó para incluir el formulario personalizado maquetado en la plantilla submission/submissionMetadataFormField.tpl
submission/submissionMetadataFormFields.tpl	Esta plantilla cumple un rol clave en la visualización de los campos del formulario de metadatos dentro del flujo de registro o edición de un envío.	Se modificó para incluir un *Hook* personalizado usado para agregar y agrupar por temática los campos del formulario para los metadatos adicionales.

https://doi.org/10.22201/dgtic.30618096e.2025.3.4.135 **Vol. 3,Núm. 4. octubre-diciembre** 2025, págs. 35 - 64

Plantilla	Descripción	Modificación
controllers/modals/submission/viewSubmissionMetadata.tpl	Esta plantilla se encarga de presentar a los usuarios con rol de revisor, la información de los metadatos del envío a dictaminar.	Se modificó para incluir un *Hook* personalizado para visualizar los valores de los metadatos adicionales.
frontend/components/registrationForm.tpl	Esta plantilla genera el formulario de registro de nuevos usuarios en el sistema.	
user/contactForm.tpl	Esta plantilla define el formulario utilizado para registrar o editar la información de contacto de un colaborador, como un autor o coautor, dentro del proceso editorial.	Se modificó para cambiar el campo del formulario que corresponde a la información de la afiliación del usuario. El campo original es un campo de texto y se reemplazó por un campo de tipo selección con los valores de todas las entidades y dependencias de la UNAM precargados, mediante un *Hook* personalizado.
user/public/publicProfileForm.tpl	Esta plantilla define el formulario que permite a un usuario autenticado editar la información de su perfil público, es decir, los datos que otros usuarios pueden ver sobre él dentro del sistema, principalmente en la interfaz pública.	
controllers/grid/users/reviewer/form/advancedSearchReviewerForm.tpl	Esta plantilla define la interfaz del formulario de búsqueda avanzada de revisores, permitiendo a los editores filtrarlos y seleccionarlos con base en diversos criterios.	Se modificó para visualizar correctamente el valor del campo afiliación, seleccionado durante el registro del usuario, mediante un *Hook* personalizado.
common/userDetails.tpl	Esta plantilla está asociada a la visualización de los detalles de un usuario dentro de la interfaz administrativa. Se utiliza para mostrar información básica del perfil de un usuario registrado dentro del sistema.	

En el Anexo C, se ilustra la diferencia entre la presentación de los datos en el formulario original, la organización lograda mediante la plantilla modificada, incorporada a través del *plugin* de tema, y fragmentos relevantes del código implementado.

https://doi.org/10.22201/dgtic.30618096e.2025.3.4.135 **Vol. 3,Núm. 4. octubre-diciembre** 2025, págs. 36 - 64

***Plugin* ruaReport**

Este *plugin* pertenece a la categoría de Reporte. Éstos son extensiones diseñadas específicamente para generar informes y exportar datos del sistema relacionados con artículos, envíos, usuarios, estadísticas, decisiones editoriales, entre otros; asimismo, permiten extraer datos estructurados en formatos como CSV, XML o JSON, facilitando su análisis, uso institucional o integración con otros sistemas. Posibilita generar informes altamente personalizados según las necesidades del equipo editorial sin depender de los reportes predefinidos del sistema.

El *plugin* fue desarrollado con el objetivo de facilitar la exportación de información relacionada con los envíos, incluyendo tanto los metadatos estándar como aquellos adicionales incorporados mediante el *plugin* ruaMetadata. Su principal funcionalidad consiste en organizar y presentar estos datos según las agrupaciones temáticas previamente definidas, lo que permite una estructuración más clara y útil para su análisis. Esta capacidad de clasificación temática mejora la comprensión del contenido por el equipo que gestiona el proceso editorial.

En el Anexo D, se muestra la figura con un ejemplo de un reporte visualizado en un programa de hoja de cálculo y fragmentos relevantes del código implementado.

***Plugin* ruaOAIDim**

Se identificó que OJS permite exponer los datos de los envíos a través del protocolo Open Archives Initiative Protocol for Metadata Harvesting (OAI-PMH). Este protocolo proporciona un mecanismo estandarizado para la recolección de metadatos desde diversas fuentes, facilitando el acceso a los mismos mediante una interfaz unificada (Tkachov, 2024). Esta funcionalidad es esencial para promover la interoperabilidad entre plataformas y sistemas de información académica.

Este *plugin* pertenece a la categoría de oaiMetadataFormats, dichos *plugins* permiten agregar formatos de metadatos que OJS puede manejar a través del protocolo OAI-PMH. Este tipo de *plugins* definen cómo será el nuevo esquema de metadatos, además de su estructura XML, espacio de nombres (namespace) y validación, extendiendo la interoperabilidad de la plataforma con distintos servicios que recolectan información de manera automatizada.

En este caso, el *plugin* que se desarrolló agrega compatibilidad con el formato DIM (DSpace Interoperability Metadata). Este formato es especialmente útil porque admite campos de metadatos más complejos y personalizados, lo que permite reflejar con mayor detalle la información que acompaña a cada envío. Gracias a esta integración, los datos adicionales que se incluyen en los recursos pueden ser correctamente expuestos mediante OAI-PMH (Das & Sutradhar, 2018). El objetivo principal de esta integración es proveer un mecanismo para que los recursos que sean aceptados puedan ser recolectados por la RUA u otras plataformas compatibles en el futuro, cumpliendo con los estándares de interoperabilidad y preservación de información académica.

En el Anexo E, se presentan figuras que ilustran la pantalla de exportación de datos en el protocolo OAI-PMH, proporcionando una visualización detallada del proceso de extracción y exposición de los metadatos a través de este protocolo, así como fragmentos relevantes del código implementado.

https://doi.org/10.22201/dgtic.30618096e.2025.3.4.135 **Vol. 3,Núm. 4. octubre-diciembre** 2025, págs. 37 - 64

6. DISEÑO Y PRUEBA DEL FLUJO DE TRABAJO

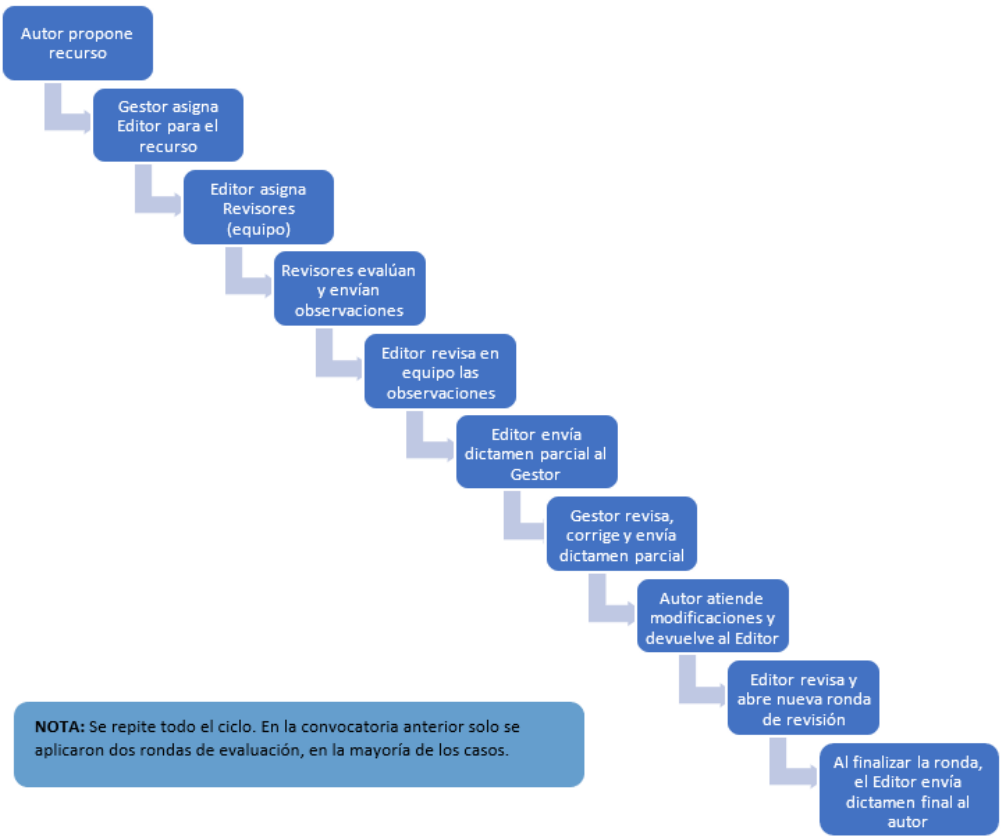
Con base en el documento redactado para guiar el trabajo de evaluación de la convocatoria 2025, con título "Informe de análisis del sistema. Open Journal Systems", en esta sección, se describe brevemente el análisis, diseño y pruebas que se llevaron a cabo para el planteamiento del flujo de trabajo en OJS, con el fin de llevar a cabo el seguimiento, evaluación y selección de recursos para publicar en la RUA.

6.1 FLUJO DE TRABAJO

El planteamiento del flujo de trabajo se llevó a cabo mientras se completaba el desarrollo necesario para incorporar los metadatos faltantes, descrito en las secciones anteriores de este documento; para ello, se realizó el análisis de las posibilidades de la plataforma en su proceso enfocado a revistas y sus similitudes con el planteado en la convocatoria y, más tarde, se ejecutaron varias etapas de pruebas con el fin de afinar iterativamente dicho flujo de trabajo dentro de OJS, el cual quedó definido como se muestra en la Figura 5 (Valenzuela, comunicación personal, 2024).

Figura 5

Flujo de trabajo OJS-RUA



https://doi.org/10.22201/dgtic.30618096e.2025.3.4.135 Vol. 3,Núm. 4. octubre-diciembre 2025, págs. 38 - 64

Dentro del flujo definido, se consideraron los siguientes roles:

- **Gestor.** Organiza todo el trabajo de evaluación de recursos, verifica y envía los dictámenes parciales y da seguimiento a todo el proceso.
- **Autor.** Realiza la propuesta de recursos para su evaluación por medio de metadatos y componentes de OJS (documentos).
- **Editor.** Verifica la llegada de los metadatos y componentes de OJS (documentos) y asigna los recursos a los revisores (es el responsable del equipo de evaluación). Coordina la evaluación y emisión de dictamen parcial y final, así como envía el dictamen final al autor.
- **Revisores.** Evalúan los recursos asignados y devuelven al editor sus observaciones en cada ronda necesaria (son los miembros del equipo de evaluación).

6.2 PRUEBAS

Se llevaron a cabo pruebas individuales durante todo el proceso de definición del flujo de trabajo en OJS y, más tarde, con el primer planteamiento ajustado, se llevaron a cabo tres ciclos de pruebas con un grupo de 7 personas que más tarde participaron en el proceso de evaluación para la convocatoria 2025.

El objetivo fue identificar las situaciones técnicas reales que se desalineaban respecto al flujo definido para la evaluación en dicha convocatoria. Este proceso permitió determinar la configuración y flujo de trabajo finales y, así, atender de manera óptima las propuestas de los docentes para su correcto seguimiento, evaluación y posterior publicación de recursos digitales aprobados en la Red Universitaria de Aprendizaje.

6.2.1 PRUEBAS INDIVIDUALES

Con el fin de plantear una primera propuesta de configuración y flujo de trabajo, se realizaron nueve flujos completos de recursos de manera individual por parte de la responsable de esta actividad. Esto permitió la definición del proceso que se sometió a las pruebas grupales, empleando las funcionalidades y recursos de OJS para asemejar lo más posible un proceso editorial a un proceso de evaluación de recursos educativos.

Roles participantes en la primera etapa

- **Autor.** Realiza una propuesta de recursos por medio del llenado de los metadatos extendidos en OJS.
- **Editor.** Verifica la información de los recursos y los asigna a los revisores.
- **Revisores.** Evalúan los recursos asignados y devuelven al editor su documento de evaluación.

El flujo original planteado fue el siguiente:

1. Asignación de editor
2. Asignación de revisor
3. Revisión

https://doi.org/10.22201/dgtic.30618096e.2025.3.4.135 **Vol. 3,Núm. 4. octubre-diciembre** 2025, págs. 39 - 64

4. Notificación al editor
5. Notificación al autor

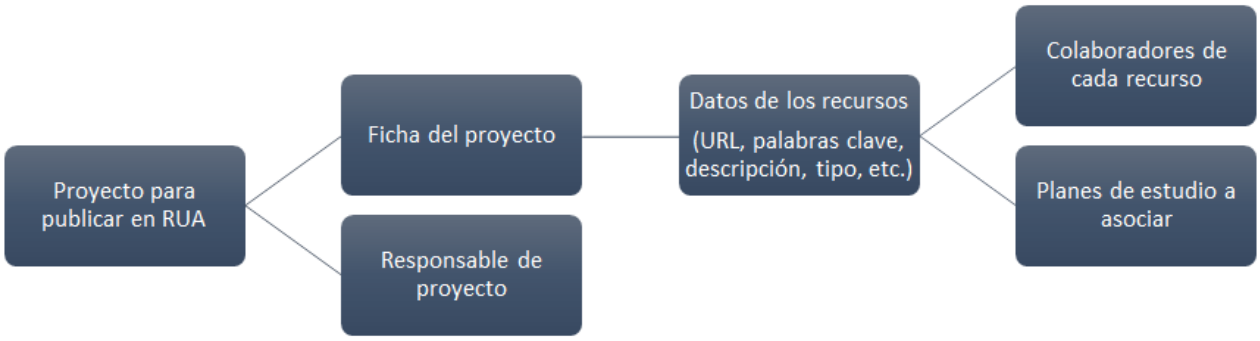
En este caso, se propuso que el editor tuviera privilegio para registrar decisiones editoriales y no se consideró la presencia de un Gestor; además, originalmente se planteó que las propuestas fueran por proyecto.

Debido a que una propuesta de artículo no tenía manera de registrar subproductos, en el esquema de envío de proyectos, se tenía que regresar al documento en Excel que antes llenaban los profesores con las especificaciones para cada recurso que lo integraba, lo cual era justamente lo que quería evitarse. Es por ello que se determinó que, a partir de ese momento, el envío de propuestas se realizaría por recurso, de manera que cada uno contara con todos los metadatos debidamente registrados.

Todas estas decisiones llevaron a la modificación del registro de propuestas en OJS que se había propuesto originalmente, el cual se muestra en la Figura 6.

Figura 6

Registro de propuestas por proyecto



El registro de propuestas por recurso se muestra en la sección de Metadatos (ver Figura 1).

6.2.2 PRUEBAS GRUPALES

Se realizaron tres sesiones con el grupo de usuarios, las cuales permitieron revisar de manera detallada el flujo de trabajo en la plataforma, así como los permisos necesarios para cada rol en todas las fases.

En este proceso, participó una gestora, una editora, tres revisores, un autor y una coordinadora de la prueba.

Primera sesión

Se describen las actividades realizadas para esta primera aproximación de los usuarios al flujo:

- Se llevó a cabo la presentación del objetivo del proyecto de automatización de la evaluación de recursos para la siguiente convocatoria de la RUA.

https://doi.org/10.22201/dgtic.30618096e.2025.3.4.135 **Vol. 3,Núm. 4. octubre-diciembre** 2025, págs. 40 - 64

- Se realizó el primer ejercicio de propuesta de un recurso para la prueba de cuentas de todos los usuarios.
- Se identificó la necesidad de tres niveles de roles, en lugar de dos que se habían considerado de inicio (Editor y Revisor): se planteó la importancia de considerar a un Gestor, además del Editor y el de los Revisores, con el fin de organizar los equipos de trabajo con un responsable y registrar todas las aportaciones de los integrantes en la plataforma, además de mantener un rol que coordinara todo el trabajo de evaluación (Gestor). Esta modificación se reflejó en el documento de flujo de trabajo (se realizaron 5 versiones con base en todos los hallazgos de las pruebas).
- Se identificó que era necesario otorgar permisos amplios a los autores debido a la necesidad de compartir con ellos los dictámenes parcial y final por medio de documentos PDF.

Segunda sesión

- Esta sesión llegó hasta la primera ronda de revisión del recurso, pues se encontró que, si un usuario inicia la segunda ronda de manera anticipada por error, ya no se puede concluir el proceso completo en la primera.

Tercera sesión

- Se explicó todo el flujo de principio a fin, considerando al Gestor y al Editor, así como la revisión de dictámenes parciales antes de su envío al Gestor.
- Se replicó dicho flujo, desde la propuesta de un nuevo recurso hasta la emisión del dictamen final, con base en el documento definitivo que se generó.
- Se identificaron los espacios idóneos de OJS para compartir documentos internos y los que se requerían entregar al autor.

7. RESULTADOS

El desarrollo e implementación de los *plugins* permitió extender la funcionalidad de la plataforma OJS, integrando en su flujo los metadatos requeridos para catalogar los recursos propuestos en la RUA. Esto permitió el uso de OJS como herramienta de gestión para las etapas de recepción, revisión y dictaminación de dichos recursos.

Uno de los principales beneficios de esta solución es que permitió garantizar un proceso automatizado, estandarizado y trazable, tanto en la gestión del proceso como en la comunicación con los autores. Además, se logró la exposición de los metadatos mediante un formato estándar de interoperabilidad, específicamente OAI-PMH, con el objetivo de facilitar la importación de los recursos aprobados al sistema de la RUA.

La versión adaptada de la plataforma se empleó en la convocatoria 2025 de la RUA. La implementación del flujo de trabajo completo dentro de la plataforma se llevó a cabo de manera satisfactoria y permitió gestionar eficientemente todas las etapas involucradas en la recepción, evaluación y dictaminación de los recursos.

https://doi.org/10.22201/dgtic.30618096e.2025.3.4.135 **Vol. 3,Núm. 4. octubre-diciembre** 2025, págs. 41 - 64

Como se aborda a lo largo del documento, la definición de dicho flujo fue el resultado de un proceso iterativo de pruebas que permitió identificar los aspectos que se requerían adaptar en la plataforma para ser utilizada en un contexto que no es aquel para el cual fue diseñada.

8. CONCLUSIONES

La adaptación de la plataforma OJS mediante el desarrollo e integración de *plugins* específicos es una solución para la automatización del proceso de evaluación de los recursos propuestos en el marco de la RUA. La incorporación de metadatos especializados y la utilización del flujo de trabajo en OJS contribuyeron a fortalecer la estandarización y eficiencia del proceso. La experiencia obtenida durante la convocatoria 2025 confirma la utilidad y pertinencia de esta implementación.

Estos desarrollos asegurarán una integración efectiva entre los flujos editoriales ofrecidos por OJS y los requerimientos particulares de catalogación y descripción didáctica que caracterizan a los recursos de la RUA, permitiendo así mantener la coherencia con su modelo de gestión de recursos educativos digitales.

Uno de los próximos pasos fundamentales consiste en migrar la funcionalidad de los *plugins* desarrollados, así como del flujo de trabajo planteado, desde OJS 3.3 hacia la versión más reciente del sistema, OJS 3.4. Esta actualización no sólo responde a la necesidad de mantener la compatibilidad y seguridad de la plataforma, sino también a los cambios estructurales significativos introducidos en esta nueva versión.

Este proceso permitirá tanto extender la vida útil de los *plugins* y aprovechar las nuevas capacidades del sistema como facilitar su mantenibilidad a largo plazo.

AGRADECIMIENTOS

Se agradece a los siguientes miembros del grupo por todas sus aportaciones al proyecto: Gestora, Gabriela Bañuelos Sandoval; Editora, María del Carmen Ramos Nava; Revisora, Laura Azucena Lira Jiménez; Revisor, Emilio José Quiroz Galván; Revisor y Asesor técnico, Alan López de Jesús y Autor, Daniel González Lorenzo.

REFERENCIAS

Asmecher. (2017). *OJS 3 custom metadata fields* [Publicación en un foro]. Public Knowledge Project Forum. https://forum.pkp.sfu.ca/t/ojs-3-custom-metadata-fields/28674

Bolboacă, A. (2021). *Practical Remote Pair Programming: Best practices, tips, and techniques for collaborating productively with distributed development teams*. Packt Publishing. https://books.google.com.mx/books?id=HiIlEAAAQBAJ

Das, A., & Sutradhar, B. (2018). Harvesting of Additional Metadata Schema into DSpace through OAI-PMH: Issues and Challenges. *SRELS Journal of Information Management*, 1-7. https://doi.org/10.17821/srels/2018/v55i1/116603

Hartstein, J., & Blümel, C. (2021). Editors between Support and Control by the Digital Infrastructure—Tracing the Peer Review Process with Data from an Editorial Management System. *Frontiers in Research Metrics and Analytics, 6*. https://doi.org/10.3389/frma.2021.747562

https://doi.org/10.22201/dgtic.30618096e.2025.3.4.135 **Vol. 3,Núm. 4. octubre-diciembre** 2025, págs. 42 - 64

Kotahi. (s. f.). *Kotahi, the future of scholar-led publishing*. About - Kotahi. Recuperado 2 de junio de 2025, de https://kotahi.community/about/

Kude, T., Mithas, S., Schmidt, C. T., & Heinzl, A. (2019). How Pair Programming Influences Team Performance: The Role of Backup Behavior, Shared Mental Models, and Task Novelty. *Information Systems Research, 30*(4), 1145-1163. https://doi.org/10.1287/isre.2019.0856

Miller, S. J. (2022). *Metadata for digital collections: A how-to-do-it manual* (Second edition). ALA Neal-Schuman.

Open Researcher and Contributor ID. (s. f.). *About ORCID*. Open Researcher and Contributor ID. https//info.orcid.org/what-is-orcid/

Public Knowledge Project. (s. f.-a). *PKP Timeline*. Simon Fraser University. Public Knowledge Project. Recuperado 30 de mayo de 2025, de https://pkp.sfu.ca/about/timeline/

Public Knowledge Project. (s. f.-b). *Theming the Editorial Backend*. Theming the Editorial Backend. Recuperado 4 de junio de 2025, de https://docs.pkp.sfu.ca/pkp-theming-guide/en/theme-backend.html

Public Knowledge Project. (s. f.-c). *OJS Workflow and Platform*. Recuperado 10 de junio de 2025, de https://pkp.sfu.ca/wp-content/uploads/2024/06/OJS-Workflow-and-Platform-2024.png

Public Knowledge Project. (s. f.-d). *Open Journal Systems MVC Diagram*. PKP Docs. Recuperado 10 de junio de 2025, de https://docs.pkp.sfu.ca/ojs-2-technical-reference/en/2_introduction.html

Public Knowledge Project. (2025). *Plugin Guide for OJS and OMP*. PKP Docs. https://docs.pkp.sfu.ca/dev/plugin-guide/3.3/en/

Tabatadze, B. (2024). Technological Aspects of Open Journal Systems (OJS). *Journal of Technical Science and Technologies, 8*(1), 23-29. https://doi.org/10.31578/jtst.v8i1.151

Tan, X., Lv, X., Jiang, J., & Zhang, L. (2024). Understanding Real-Time Collaborative Programming: A Study of Visual Studio Live Share. *ACM Transactions on Software Engineering and Methodology, 33*(4), 1-28. https://doi.org/10.1145/3643672

Valenzuela, R. (en prensa) Red Universitaria de Aprendizaje: evolución del proyecto en la UNAM en *Recursos Educativos Abiertos 3D en la enseñanza superior y el Aprendizaje Activo en entornos híbridos*. Universidad Nacional Autónoma de México.

Wang, W. (2022). Research on Using Docker Container Technology to Realize Rapid Deployment Environment on Virtual Machine. *2022 8th Annual International Conference on Network and Information Systems for Computers (ICNISC)*, 541-544. https://doi.org/10.1109/ICNISC57059.2022.00112

Willinsky, J. (2005). Open Journal Systems: An example of open source software for journal management and publishing. *Library Hi Tech, 23*(4), 504-519. https://doi.org/10.1108/07378830510636300

https://doi.org/10.22201/dgtic.30618096e.2025.3.4.135 **Vol. 3,Núm. 4. octubre-diciembre** 2025, págs. 43 - 64

ANEXO A. ANÁLISIS DE METADATOS RUA DISPONIBLES EN OJS

A continuación, se presenta el desglose detallado de los metadatos requeridos para la RUA. Con el objetivo de facilitar su identificación, se han marcado con las siglas OJS aquellos elementos que se localizaron en la plataforma durante el análisis descrito en el cuerpo principal del documento. Este anexo tiene como finalidad complementar dicho análisis, ofreciendo la visión comparativa que permitió identificar coincidencias, vacíos y oportunidades de alineación entre los requerimientos de la RUA y la estructura original de metadatos disponible en OJS.

NOTA: Se indican con el texto "*Incluido en OJS*" aquellos campos que, inicialmente, se identificaron como existentes, aunque no estuvieran nombrados exactamente como se requería en la RUA.

Ficha del proyecto

Tipo de proyecto

Número de proyecto PAPIME, PAPIIT o INFOCAB

Nombre oficial de proyecto ***Incluido en OJS***

Responsable de proyecto

Nombre completo del solicitante ***Incluido en OJS***

Correo responsable ***Incluido en OJS***

Entidad de adscripción ***Incluido en OJS***

Datos de los recursos

Título del recurso ***Incluido en OJS***

Localización (URL) o enlace al archivo o carpeta del recurso (Google Drive, OneDrive, Dropbox, etc.) ***Incluido en OJS***

Palabras clave (3) ***Incluido en OJS***

Descripción (general) ***Incluido en OJS***

Tipo de recurso

Descripción del uso educativo

Nivel educativo (bachillerato, licenciatura, posgrado)

Categoría (estudiante/profesor)

Derechos de autor. Descripción. ***Incluido en OJS***

Cita en formato APA

Número de ISBN o ISSN (si se cuenta con este dato)

Recurso modificable (Sí/No)

https://doi.org/10.22201/dgtic.30618096e.2025.3.4.135 **Vol. 3,Núm. 4. octubre-diciembre** 2025, págs. 44 - 64

Colaboradores del recurso

Mención de responsabilidad (título, nombre, tipo - autor, compilador, coordinador, editor, etc.) ***Incluido en OJS***

Plan de estudios

Plantel donde se imparte

Nombre del plan estudios

Año del plan de estudios

Semestre / Año escolar

Asignatura (nombre y clave)

Sitio de referencia del plan de estudios "oficial" y los programas de las asignaturas (URL)

Documentos adicionales

Carta de cesión de derechos - Componente ***Incluido en OJS***

Carta aval (en algunos casos) - Componente ***Incluido en OJS***

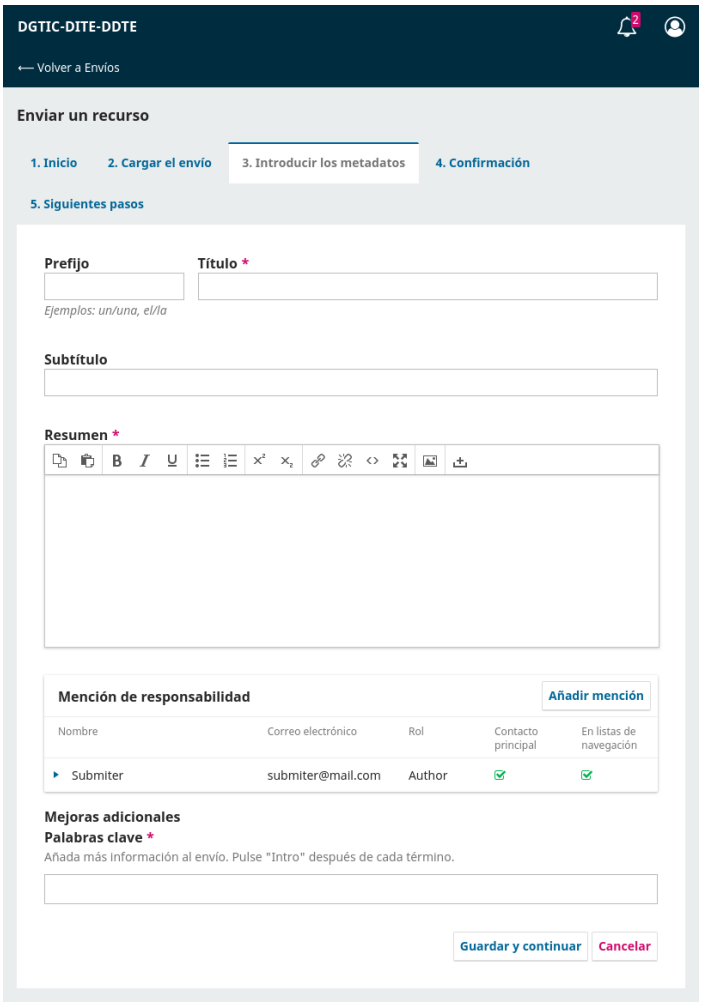
ANEXO B. COMPARATIVAS DE PANTALLAS DE OJS CON EL *PLUGIN* RUAMETADATA INSTALADO Y HABILITADO

A continuación, se presentan dos imágenes que permiten comparar el formulario de registro de un nuevo envío antes y después de la incorporación del *plugin* ruaMetadata. La Figura 7 muestra la versión original del formulario con los metadatos que utiliza OJS por defecto. En la Figura 8, se observa el formulario modificado tras la activación del *plugin*, el cual introduce nuevos campos para capturar metadatos adicionales.

https://doi.org/10.22201/dgtic.30618096e.2025.3.4.135 Vol. 3,Núm. 4. octubre-diciembre 2025, págs. 45 - 64

Figura 7

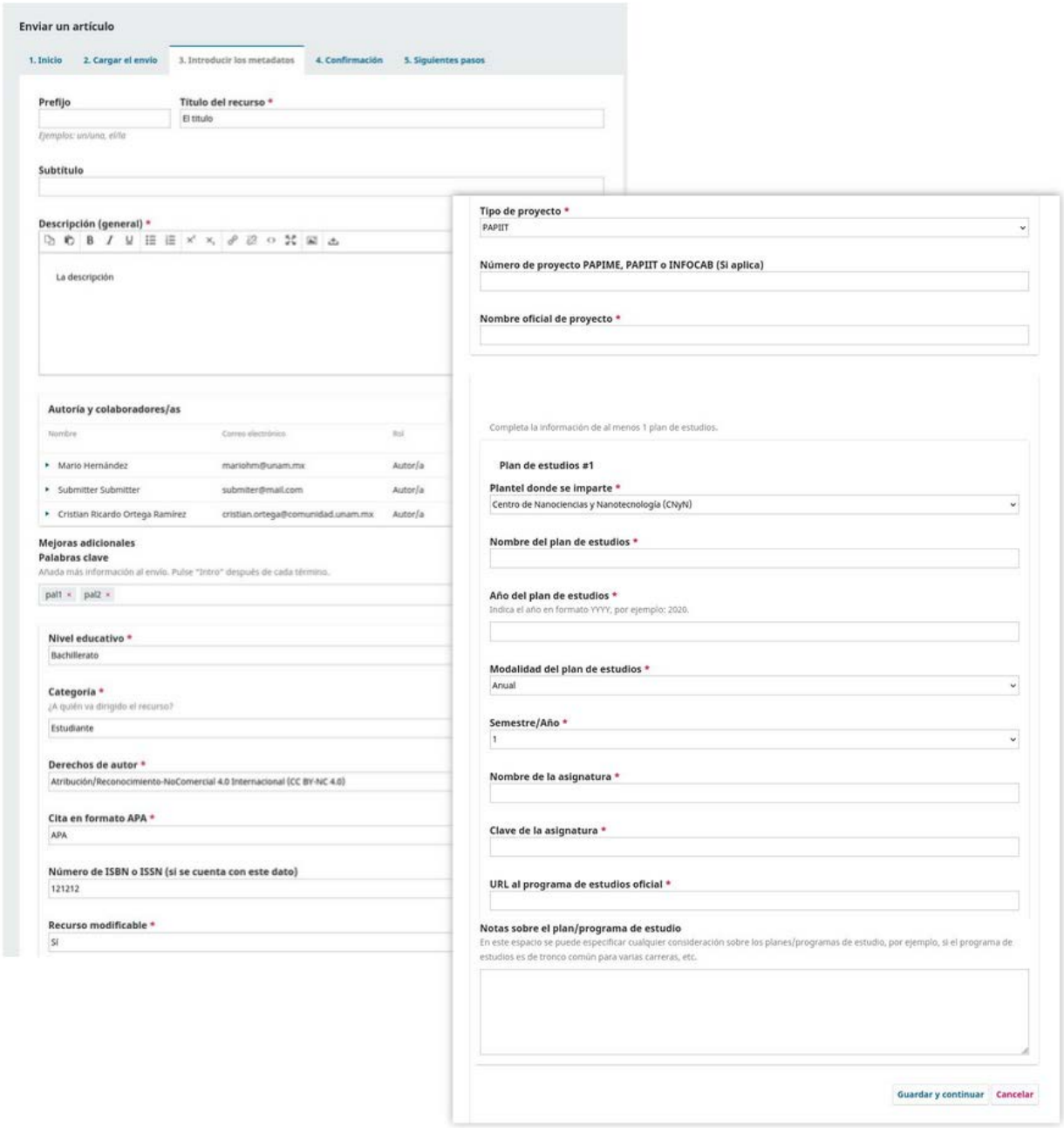
Formulario original de registro de un nuevo envío



https://doi.org/10.22201/dgtic.30618096e.2025.3.4.135 **Vol. 3,Núm. 4. octubre-diciembre** 2025, págs. 46 - 64

Figura 8

Formulario de registro de un nuevo envío modificado

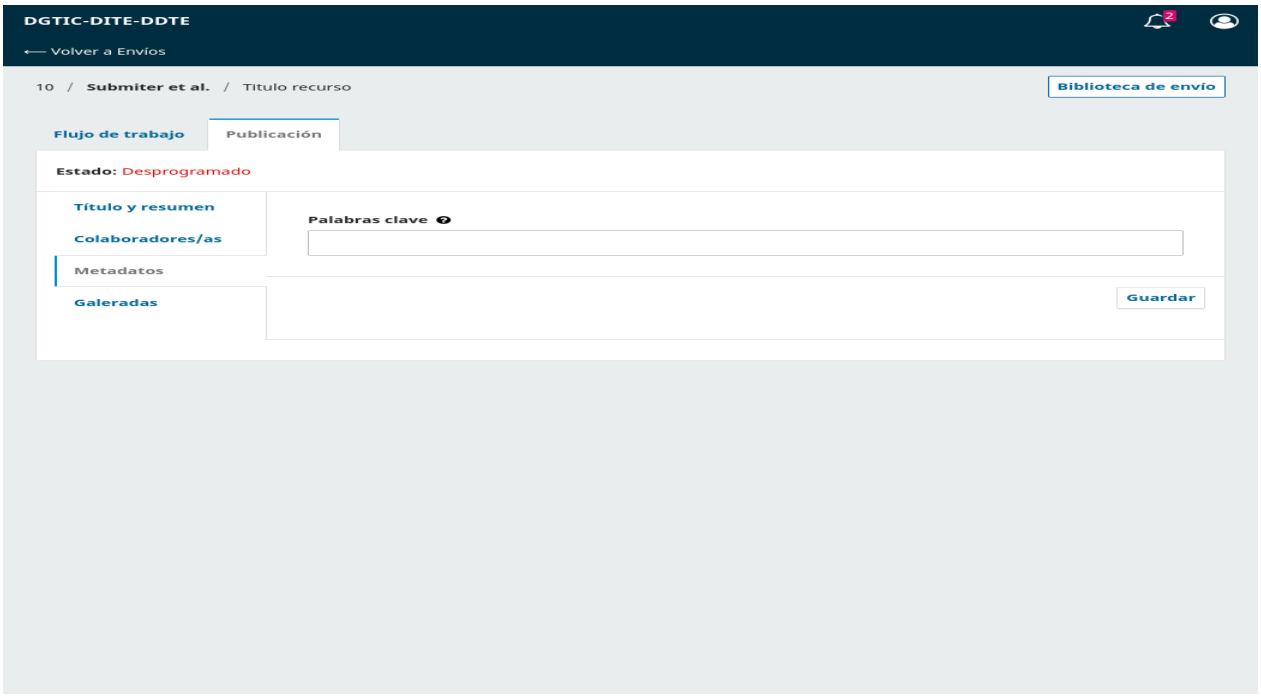


De manera complementaria, las Figuras 9 y 10 muestran el formulario de edición de un envío antes y después de la incorporación del *plugin* ruaMetadata. En la Figura 9, se observa la versión original del formulario y, en la Figura 10, se muestra el formulario ampliado tras la activación del *plugin*.

https://doi.org/10.22201/dgtic.30618096e.2025.3.4.135 Vol. 3,Núm. 4. octubre-diciembre 2025, págs. 47 - 64

Figura 9

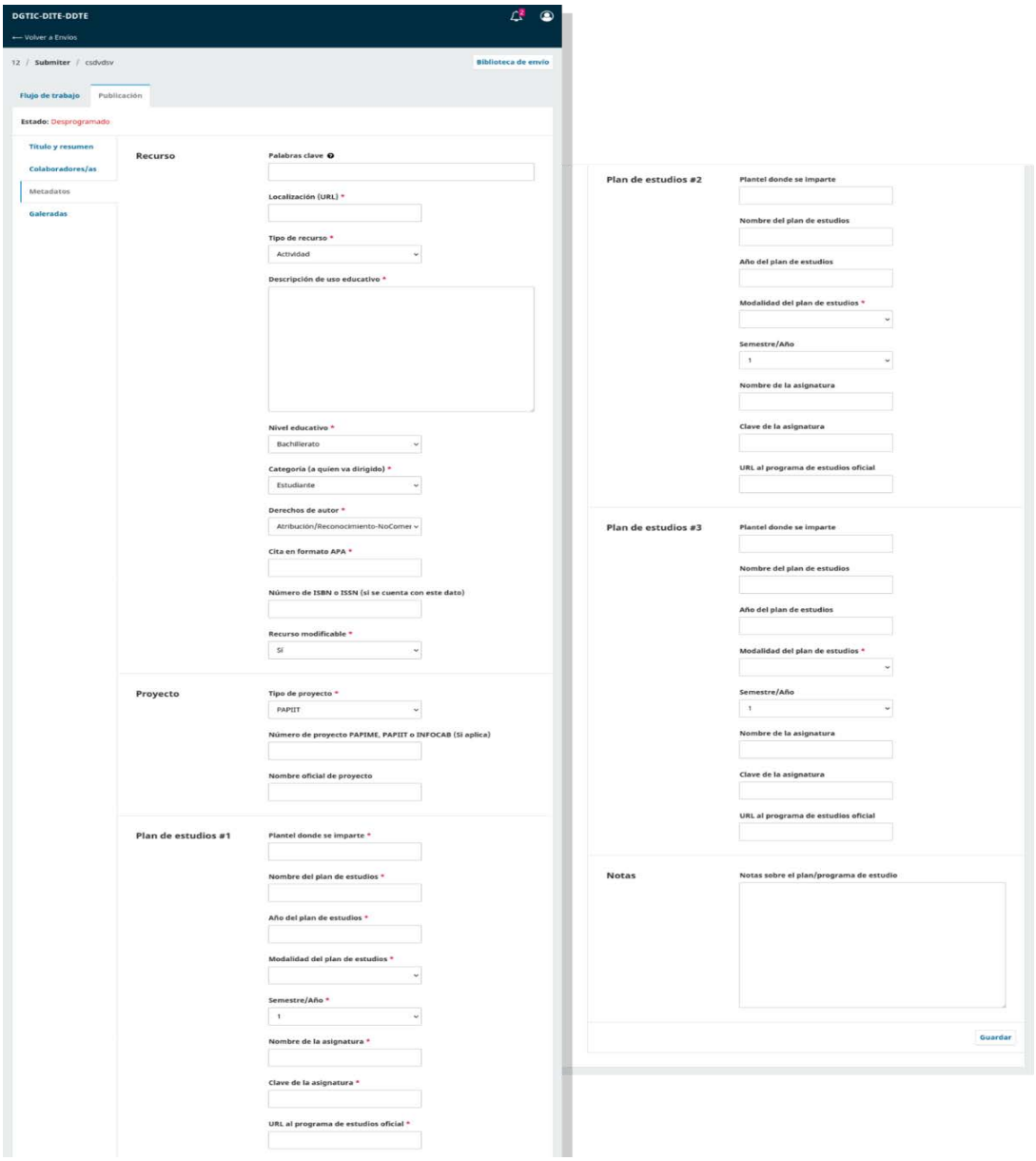
Formulario original de edición de un envío



https://doi.org/10.22201/dgtic.30618096e.2025.3.4.135 Vol. 3,Núm. 4. octubre-diciembre 2025, págs. 48 - 64

Figura 10

Formulario de edición de un envío modificado



La Figura 11 muestra fragmentos del código fuente del *plugin* que realizan lo siguiente:

1. Registrar hooks.
2. Agregar campos al esquema del envío.
3. Hacer disponible la información en la plantilla.

https://doi.org/10.22201/dgtic.30618096e.2025.3.4.135 Vol. 3,Núm. 4. octubre-diciembre 2025, págs. 49 - 64

Figura 11

Fragmentos de código del plugin ruaMetadata

```php
<?php
import('lib.pkp.classes.plugins.GenericPlugin');

class RuaMetadataPlugin extends GenericPlugin
{
   const RUA_METADATA = [
      'nivel_educativo' => [
         'schema' => [
            'type' => 'string',
            'multilingual' => false,
            'apiSummary' => true,
            'validation' => ['required']
         ],
         'form' => [
            'clazz' => '\PKP\components\forms\FieldSelect',
            'group' => 'Recurso',
            'args' => [
               'label' => 'Nivel educativo',
               'options' => self::NIVEL_EDUCATIVO,
               'value' => '',
               'isRequired' => true
            ]
         ]
      ],
      // ...[otros metadatos similares]...
   ];

1  function register($category, $path, $mainContextId = null)
   {
      $success = parent::register($category, $path, $mainContextId);
      if ($success && $this->getEnabled($mainContextId)) {
         /** Hooks para el envío **/
         HookRegistry::register(
           'Schema::get::submission',
           array($this, 'addToSchema')
         );
         HookRegistry::register(
        'Templates::Submission::SubmissionMetadataForm::AdditionalMetadata',
         array($this, 'addFormFields')
         );
         // ... [otros hooks] ...
         $this->_registerTemplateResource();
      }
      return $success;
   }

2  public function addToSchema($hookName, $args)
   {
      $schema = $args[0];
      foreach (self::RUA_METADATA as $key => $value) {
         $schema->properties->{$key} = (object) $value['schema'];
      }
      return false;
   }

3  public function addFormFields($hookName, $args)
   {
      $smarty =& $args[1];
      $output =& $args[2];
      $smarty->assign('nivelEducativo', self::NIVEL_EDUCATIVO);
      // ... [asignaciones similares para otros metadatos] ...
      $output .= $smarty->fetch(
        $this->getTemplateResource('sectionFormAdditionalFields.tpl')
      );
      return false;
   }
   //... [otros métodos] ...
}
```

https://doi.org/10.22201/dgtic.30618096e.2025.3.4.135 **Vol. 3,Núm. 4. octubre-diciembre** 2025, págs. 50 - 64

La Figura 12 muestra un fragmento del código fuente de la plantilla personalizada que permite incluir los nuevos metadatos.

Figura 12

Fragmento de código de la plantilla personalizada (sectionFormAdditionalFields.tpl).

```
<div  class="section pkp_controllers_grid">

    {** ... Contenido del template original ... **}

    <div class="section">
        {fbvFormSection title="Nivel educativo" for="nivel_educativo" translate=false
        required=true required=true }
            {fbvElement required=true type="select" from=$nivelEducativo
            id="nivel_educativo" translate=false selected=$nivel_educativo}
        {/fbvFormSection}
    </div>

    <div class="section">
        {fbvFormSection title="Categoría" for="categoria" translate=false required=true }
            <label class="description">¿A quién va dirigido el recurso?</label>
            {fbvElement required=true type="select" from=$categoria id="categoria"
            translate=false selected=$categoria}
        {/fbvFormSection}
    </div>

    {** ... más campos personalizados ... **}

</div>
```

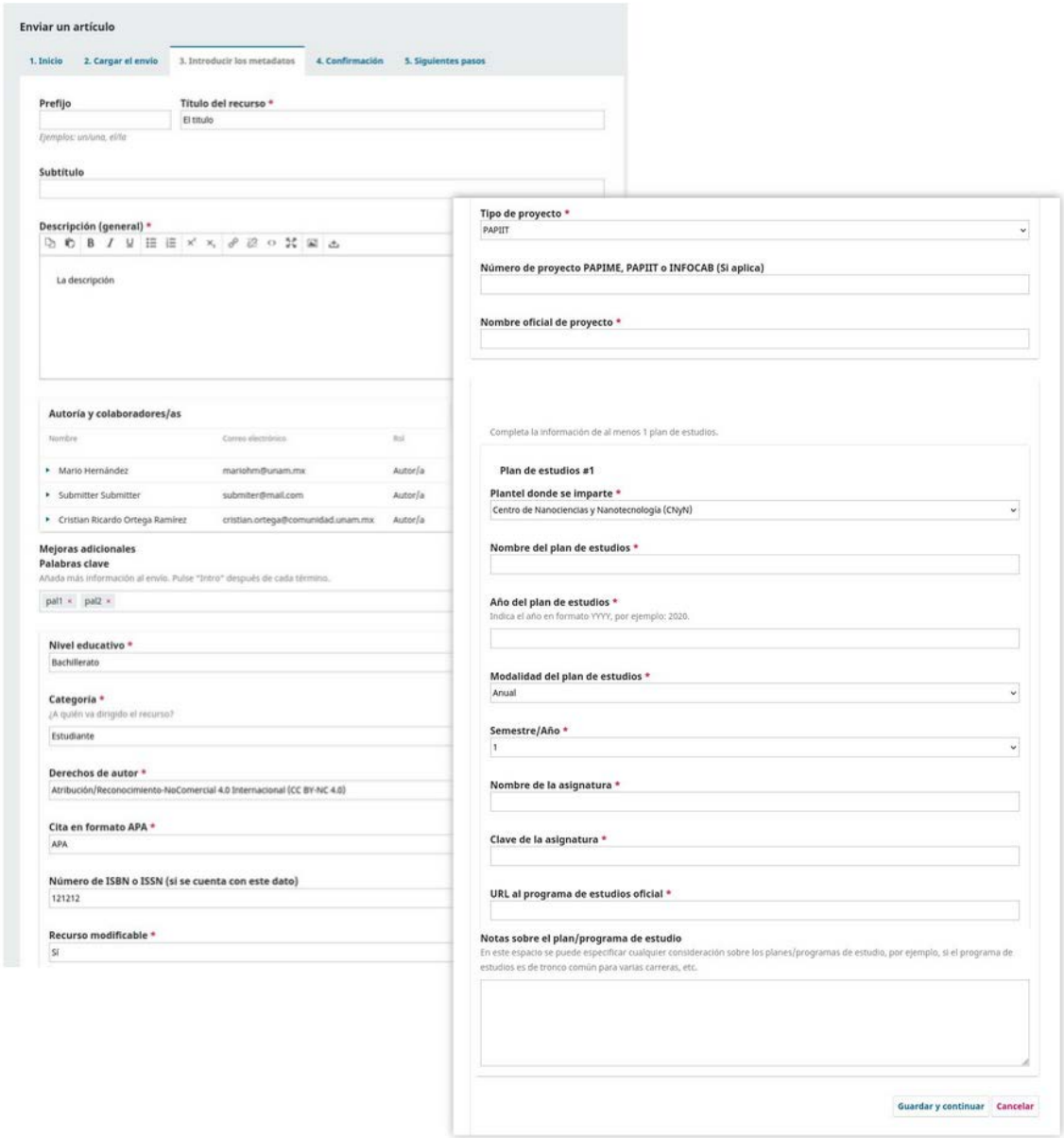
ANEXO C. COMPARATIVAS DE PANTALLAS DE OJS CON EL *PLUGIN* RUATHEME INSTALADO Y HABILITADO

A continuación, las Figuras 13 y 14 presentan una comparación entre dos versiones del formulario de nuevo envío, ambas con mejoras derivadas de la incorporación de *plugins* específicos. En la Figura 13, se muestra el formulario con el *plugin* ruaMetadata activo, el cual añade nuevos campos destinados a enriquecer la descripción del envío mediante metadatos adicionales. En la Figura 14, además de contar con dichos campos, se observa la aplicación del *plugin* ruaTheme, cuya función principal es reorganizar y agrupar los elementos del formulario en secciones temáticas. Esta estructuración mejora la claridad visual y facilita la navegación del usuario durante el proceso de registro de un envío.

https://doi.org/10.22201/dgtic.30618096e.2025.3.4.135 **Vol. 3,Núm. 4. octubre-diciembre** 2025, págs. 51 - 64

Figura 13

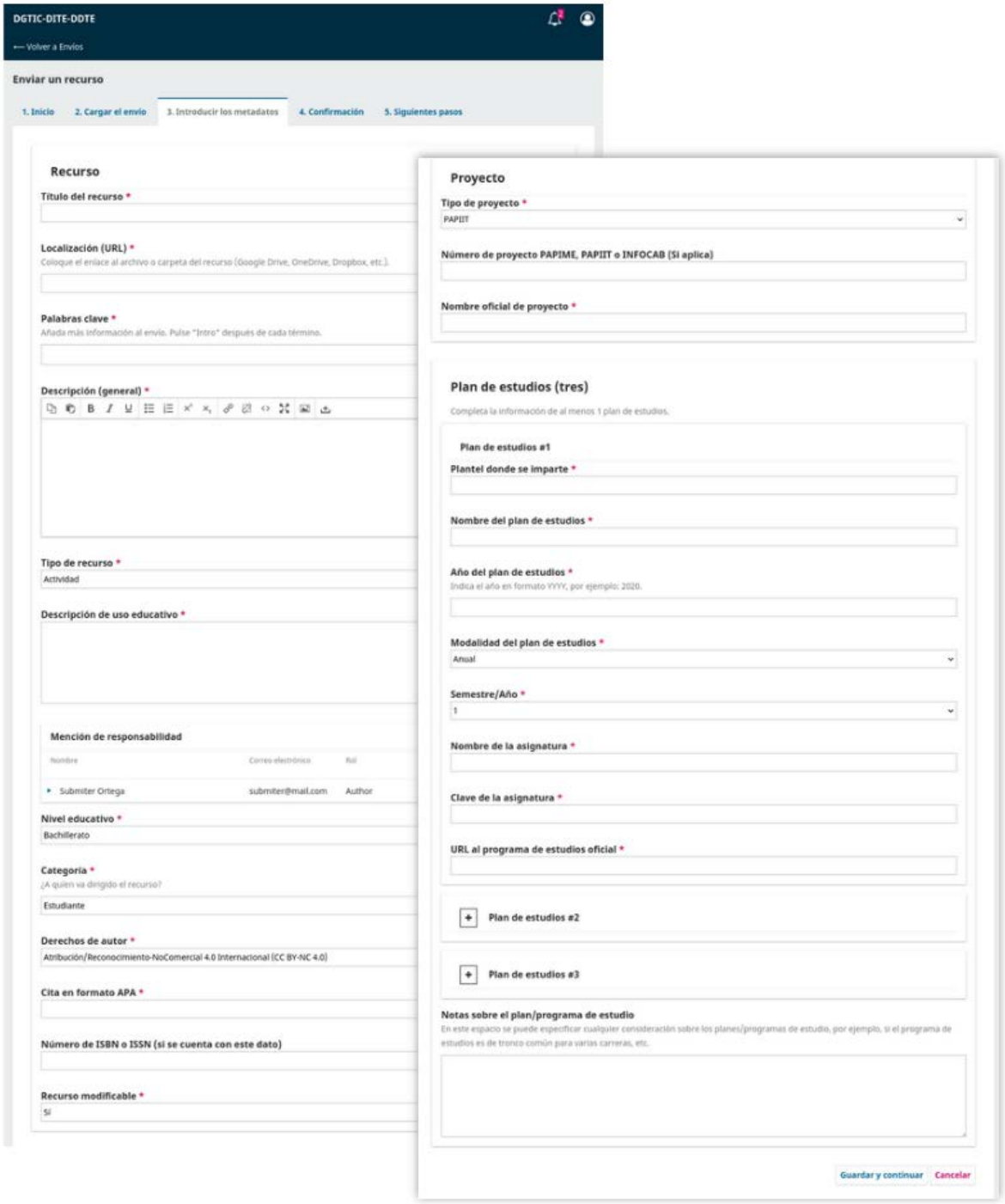
Formulario de registro de un nuevo envío con el plugin ruaMetadata



https://doi.org/10.22201/dgtic.30618096e.2025.3.4.135 Vol. 3,Núm. 4. octubre-diciembre 2025, págs. 52 - 64

Figura 14

Formulario de registro de un nuevo "envío" con los campos ordenados y agrupados



https://doi.org/10.22201/dgtic.30618096e.2025.3.4.135 **Vol. 3,Núm. 4. octubre-diciembre** 2025, págs. 53 - 64

Por otra parte, las Figuras 15 y 16 permiten comparar la ventana modal que se presenta al revisor para visualizar los metadatos de un envío, antes y después de la activación de los *plugins* ruaMetadata y ruaTheme. La Figura 15 muestra la versión original y la Figura 16 presenta la versión modificada, en la que se incluyen los nuevos metadatos definidos por ruaMetadata y se aplica la organización en secciones provista por ruaTheme. Esta nueva estructura facilita una lectura más ordenada y comprensible de la información relevante para la revisión del envío.

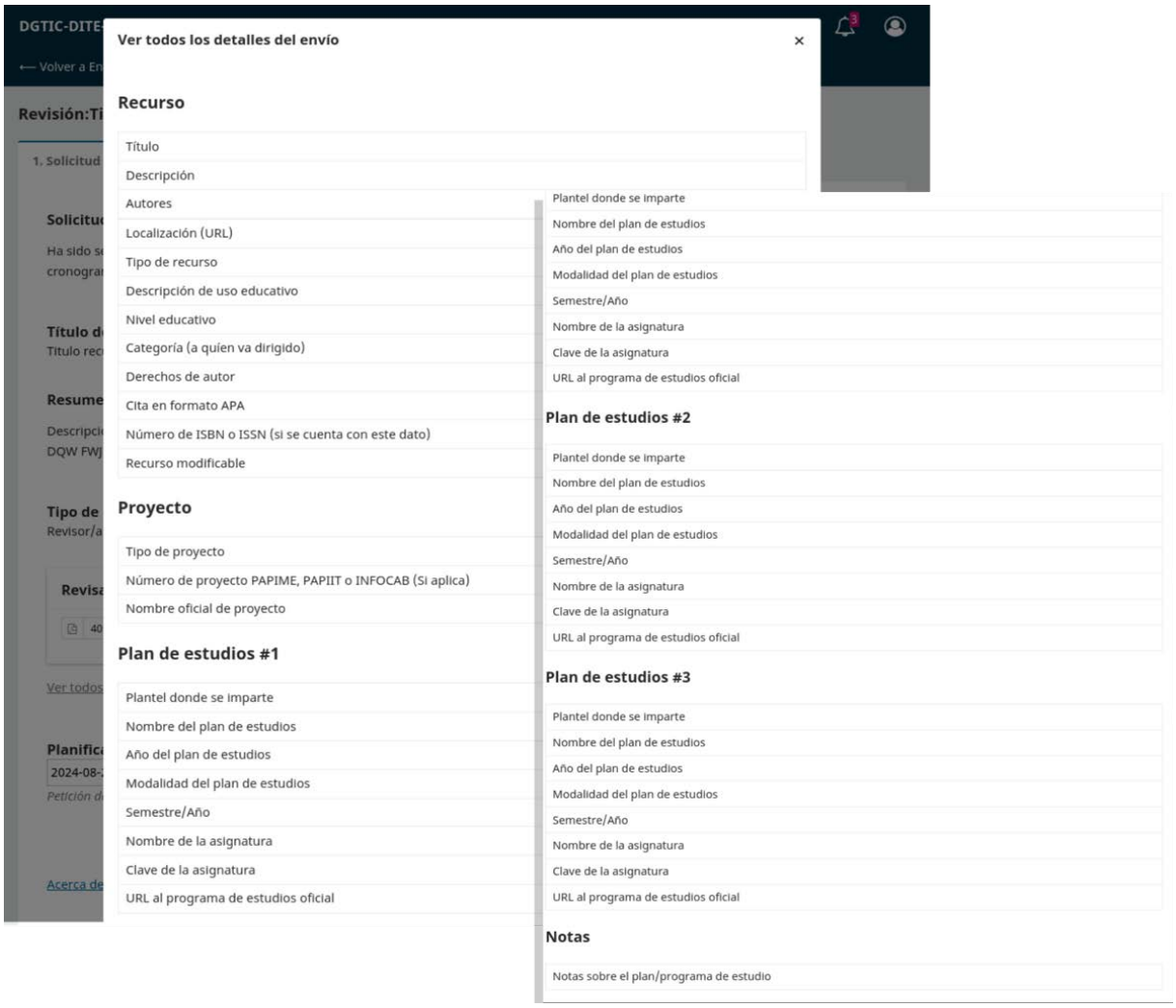
Figura 15

Ventana modal original

https://doi.org/10.22201/dgtic.30618096e.2025.3.4.135 **Vol. 3,Núm. 4. octubre-diciembre** 2025, págs. 54 - 64

Figura 16

Ventana modal con los plugins activados

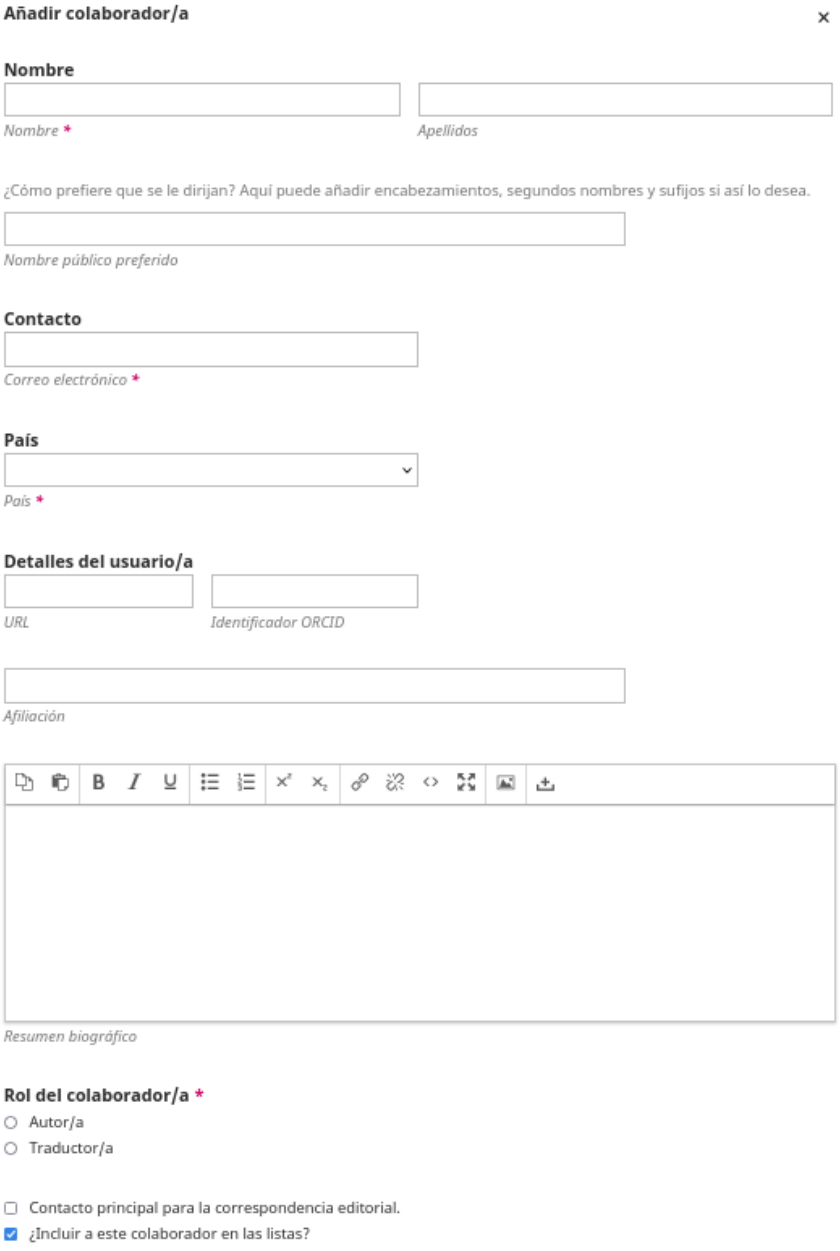


Para concluir con esta serie de comparativas, las Figuras 17 y 18 ilustran la transformación de un campo utilizado para capturar la afiliación institucional de un usuario o autor en distintos formularios del sistema. Originalmente, como se muestra en la Figura 17, dicho campo era de tipo texto libre, lo que daba lugar a inconsistencias al capturar la información. En la Figura 18, se presenta la versión modificada del campo, que ha sido reemplazado por un selector desplegable con una lista precargada de entidades y dependencias de la UNAM. Este cambio se refleja de manera uniforme en tres formularios clave: el de registro de un nuevo usuario, el de edición de perfil y el de registro de colaborador, contribuyendo a una captura más precisa, coherente y controlada de la afiliación institucional.

https://doi.org/10.22201/dgtic.30618096e.2025.3.4.135 **Vol. 3,Núm. 4. octubre-diciembre** 2025, págs. 55 - 64

Figura 17

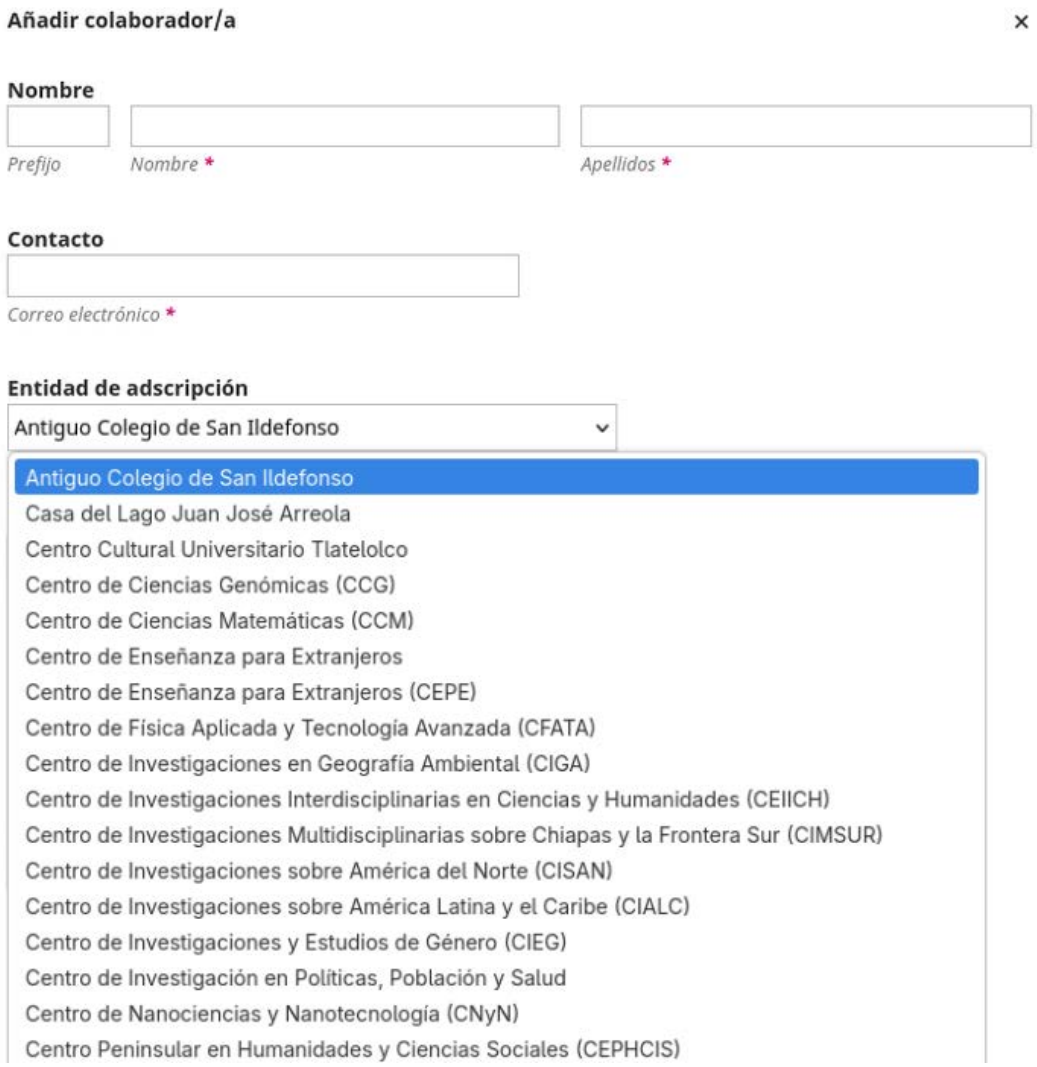
Formulario de registro de colaborador original



https://doi.org/10.22201/dgtic.30618096e.2025.3.4.135 **Vol. 3,Núm. 4. octubre-diciembre** 2025, págs. 56 - 64

Figura 18

Formulario de registro de colaborador con el campo modificado

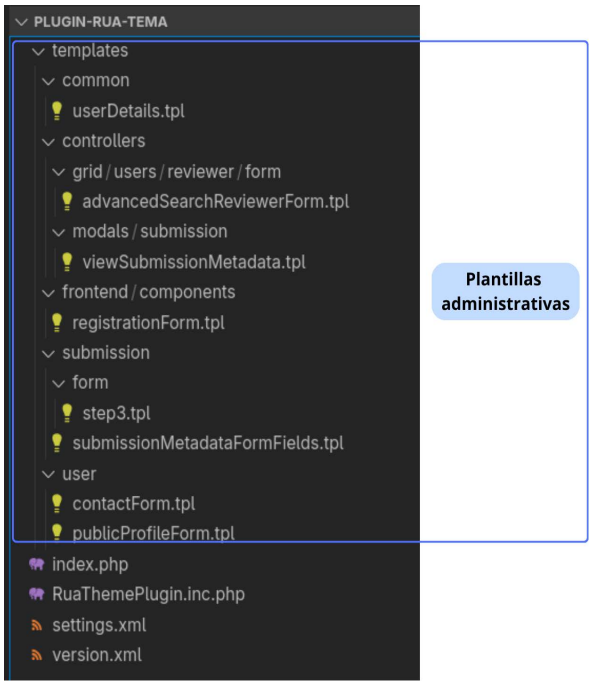


En la Figura 19, se muestra la estructura de archivos del *plugin* de tipo tema en donde se aprecian las plantillas administrativas que se sobreescriben mediante el *plugin*.

https://doi.org/10.22201/dgtic.30618096e.2025.3.4.135 **Vol. 3,Núm. 4. octubre-diciembre** 2025, págs. 57 - 64

Figura 19

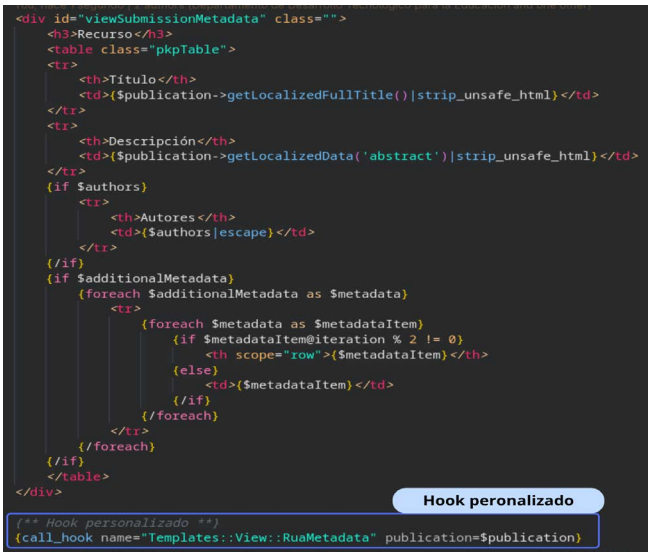
Estructura de archivos del plugin ruaTheme



En la Figura 20, se muestra el ejemplo de una plantilla adaptada para incluir un *Hook* personalizado.

Figura 20

Código de la plantilla adaptada (viewSubmissionMetadata.tpl)




```
<div id="viewSubmissionMetadata" class="">
    <h3>Recurso</h3>
    <table class="pkpTable">
    <tr>
        <th>Título</th>
        <td>{$publication->getLocalizedFullTitle()|strip_unsafe_html}</td>
    </tr>
    <tr>
        <th>Descripción</th>
        <td>{$publication->getLocalizedData('abstract')|strip_unsafe_html}</td>
    </tr>
    {if $authors}
        <tr>
            <th>Autores</th>
            <td>{$authors|escape}</td>
        </tr>
    {/if}
    {if $additionalMetadata}
        {foreach $additionalMetadata as $metadata}
            <tr>
                {foreach $metadata as $metadataItem}
                    {if $metadataItem@iteration % 2 != 0}
                        <th scope="row">{$metadataItem}</th>
                    {else}
                        <td>{$metadataItem}</td>
                    {/if}
                {/foreach}
            </tr>
        {/foreach}
    {/if}
    </table>
</div>

{** Hook personalizado **}
{call_hook name="Templates::View::RuaMetadata" publication=$publication}
```

https://doi.org/10.22201/dgtic.30618096e.2025.3.4.135 **Vol. 3,Núm. 4. octubre-diciembre** 2025, págs. 58 - 64

En la Figura 21, se muestra el código fuente del *plugin* ruaTheme.

Figura 21

Código fuente del plugin ruaTheme

```php
<?php

import('lib.pkp.classes.plugins.ThemePlugin');

class RuaThemePlugin extends ThemePlugin {

    public function isActive() {
        if (defined('SESSION_DISABLE_INIT')) return true;
        return parent::isActive();
    }
    public function init() {
        $this->setParent('defaultthemeplugin');
    }

    function getDisplayName() {
        return 'RUA theme plugin';
    }

    function getDescription() {
        return 'Tema que aplica cambios específicos para la captura y
                despliegue de los metadatos para la RUA.';
    }
}
```

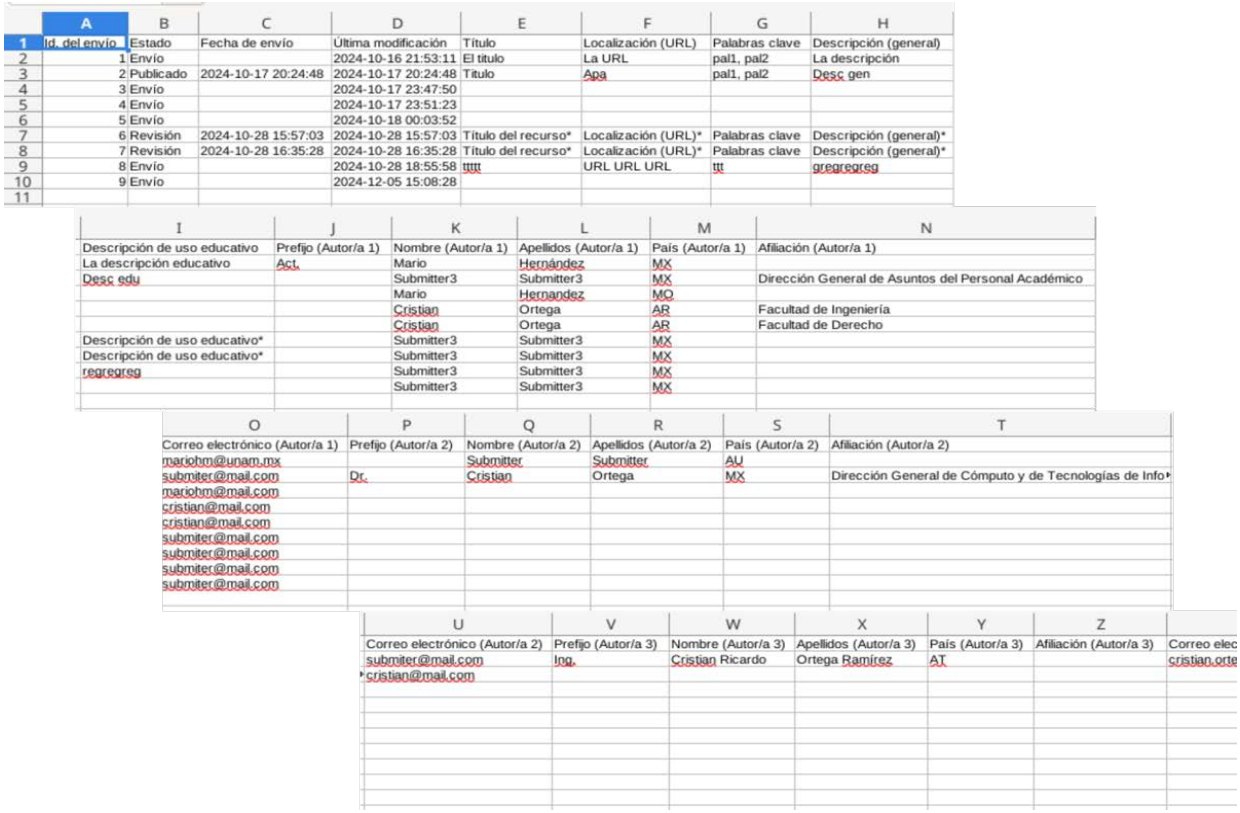
ANEXO D. EJEMPLO DE REPORTE GENERADO CON EL *PLUGIN* RUAREPORT

La Figura 22 muestra un ejemplo de un archivo en formato CSV generado a partir de la exportación de metadatos de los envíos registrados en OJS mediante el *plugin*. Cabe destacar que, además de los campos estándar, el archivo incluye los metadatos adicionales incorporados mediante el *plugin* ruaMetadata.

https://doi.org/10.22201/dgtic.30618096e.2025.3.4.135 **Vol. 3,Núm. 4. octubre-diciembre** 2025, págs. 59 - 64

Figura 22

Visualización del documento CSV con la información de los envíos



	A	B	C	D	E	F	G	H
1	Id. del envío	Estado	Fecha de envío	Última modificación	Título	Localización (URL)	Palabras clave	Descripción (general)
2	1	Envío		2024-10-16 21:53:11	El titulo	La URL	pal1, pal2	La descripción
3	2	Publicado	2024-10-17 20:24:48	2024-10-17 20:24:48	Titulo	Apa	pal1, pal2	Desc gen
4	3	Envío		2024-10-17 23:47:50				
5	4	Envío		2024-10-17 23:51:23				
6	5	Envío		2024-10-18 00:03:52				
7	6	Revisión	2024-10-28 15:57:03	2024-10-28 15:57:03	Título del recurso*	Localización (URL)*	Palabras clave	Descripción (general)*
8	7	Revisión	2024-10-28 16:35:28	2024-10-28 16:35:28	Título del recurso*	Localización (URL)*	Palabras clave	Descripción (general)*
9	8	Envío		2024-10-28 18:55:58	ttttt	URL URL URL	ttt	gregregreg
10	9	Envío		2024-12-05 15:08:28				
11								

	I	J	K	L	M	N
1	Descripción de uso educativo	Prefijo (Autor/a 1)	Nombre (Autor/a 1)	Apellidos (Autor/a 1)	País (Autor/a 1)	Afiliación (Autor/a 1)
2	La descripción educativo	Act.	Mario	Hernández	MX	
3	Desc edu		Submitter3	Submitter3	MX	Dirección General de Asuntos del Personal Académico
4			Mario	Hernandez	MO	
5			Cristian	Ortega	AR	Facultad de Ingeniería
6			Cristian	Ortega	AR	Facultad de Derecho
7	Descripción de uso educativo*		Submitter3	Submitter3	MX	
8	Descripción de uso educativo*		Submitter3	Submitter3	MX	
9	regregreg		Submitter3	Submitter3	MX	
10			Submitter3	Submitter3	MX	
11						

	O	P	Q	R	S	T
1	Correo electrónico (Autor/a 1)	Prefijo (Autor/a 2)	Nombre (Autor/a 2)	Apellidos (Autor/a 2)	País (Autor/a 2)	Afiliación (Autor/a 2)
2	mariohm@unam.mx		Submitter	Submitter	AU	
3	submiter@mail.com	Dr.	Cristian	Ortega	MX	Dirección General de Cómputo y de Tecnologías de Info
4	mariohm@mail.com					
5	cristian@mail.com					
6	cristian@mail.com					
7	submiter@mail.com					
8	submiter@mail.com					
9	submiter@mail.com					
10	submiter@mail.com					
11						

	U	V	W	X	Y	Z
1	Correo electrónico (Autor/a 2)	Prefijo (Autor/a 3)	Nombre (Autor/a 3)	Apellidos (Autor/a 3)	País (Autor/a 3)	Afiliación (Autor/a 3)
2	submiter@mail.com	Ing.	Cristian Ricardo	Ortega Ramírez	AT	
3	cristian@mail.com					

En la Figura 23, se muestran fragmentos del código fuente del *plugin* ruaReport.

https://doi.org/10.22201/dgtic.30618096e.2025.3.4.135 Vol. 3,Núm. 4. octubre-diciembre 2025, págs. 60 - 64

Figura 23

Fragmentos del código fuente del plugin ruaReport

```php
<?php

import('lib.pkp.classes.plugins.ReportPlugin');

class RUAReportPlugin extends ReportPlugin {

    function register($category, $path, $mainContextId = null) {
        $success = parent::register($category, $path, $mainContextId);
        $this->addLocaleData();
        return $success;
    }

    // ...

    function display($args, $request) {
        // ...
        while ($submission = $submissions->next()) {
            // ...
            // Recupera los metadatos del envío
            $results[] = [
                'submissionId' => $submission->getId(),
                'nivel_educativo' => self::NIVEL_EDUCATIVO[
                    $submission->getData('nivel_educativo')
                ],
                'categoria' => self::CATEGORIA[
                  $submission->getData('categoria')
                ],
                //... [recupera el resto de los metadatos] ...
            ];
        }

        $columns = [
            //Encabezados envío
            __('article.submissionId'),
            // ...
        ];

        $columns = array_merge($columns, [
            'Nivel educativo',
            'Categoría',
            // ... [agrega el resto de los encabezados de los metadatos] ...
        ]);

        // ...
        // Se convierten los resultados en filas CSV
        foreach ($results as $result) {
            $row = [];
            foreach ($result as $column => $value) switch ($column) {
                // ...
            }
            fputcsv($fp, $row);
        }

        fclose($fp);
    }
    // ...
}
```

https://doi.org/10.22201/dgtic.30618096e.2025.3.4.135 **Vol. 3,Núm. 4. octubre-diciembre** 2025, págs. 61 - 64

ANEXO E. COMPARATIVAS DE PANTALLAS DE LOS DATOS EXPUESTOS MEDIANTE OAI-PMH CON EL *PLUGIN* RUAOAIDIM INSTALADO Y HABILITADO

Las Figuras 24 y 25 muestran la forma en que OJS expone los metadatos de los envíos mediante el protocolo OAI-PMH, utilizando dos formatos distintos. La Figura 24 presenta la estructura generada con el formato estándar Dublin Core (DC), el cual ofrece una representación básica y general de los metadatos. Por otra parte, la Figura 25 muestra la salida obtenida mediante el *plugin* ruaOAIDim, que permite la exposición de los metadatos en el formato DSpace Intermediate Metadata (DIM), el cual admite una mayor granularidad y organización de los elementos descriptivos.

Figura 24

OAI-PMH con el formato DC

OAI Record: oai:ojs2. <hostname> :article/2

OAI Record Header

OAI Identifier	oai:ojs2.: <hostname> :article/2 oai_dc formats
Datestamp	2025-03-05T19:30:17Z
setSpec	dgtic:ART Identifiers Records

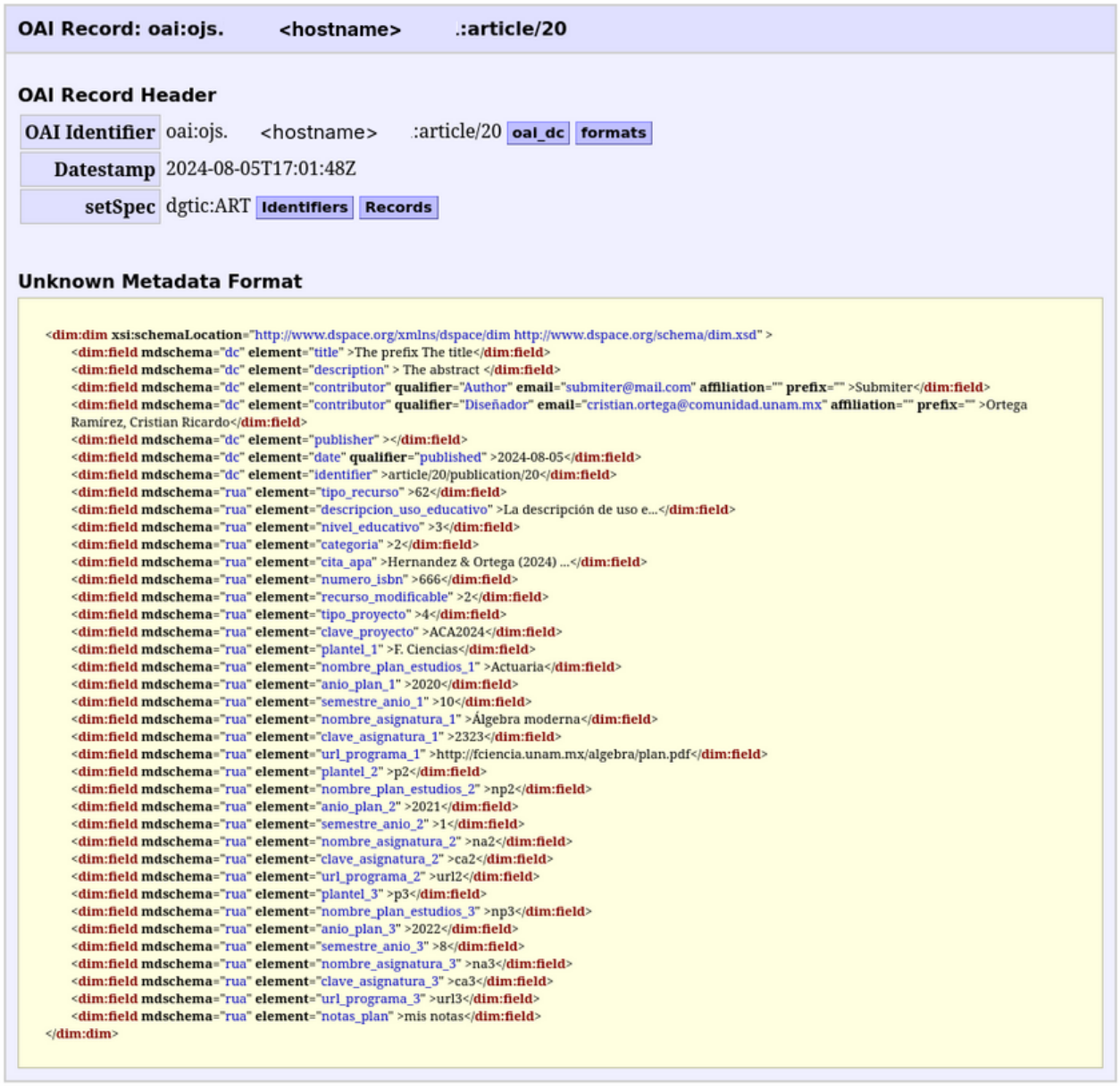
Dublin Core Metadata (oai_dc)

Title	Titulo
Author or Creator	Submitter3, Submitter3
Author or Creator	Ortega, Cristian
Subject and Keywords	pal1
Subject and Keywords	pal2
Description	Desc gen
Publisher	DGTIC
Date	2025-03-05
Resource Type	info:eu-repo/semantics/article
Resource Type	info:eu-repo/semantics/publishedVersion
Resource Type	Artículo revisado por pares
Resource Identifier	http:// <hostname> /index.php/dgtic/article/view/2
Source	DGTIC; Vol. 1 Núm. 1 (2024): RUA
Rights Management	Derechos de autor 2025 DGTIC

https://doi.org/10.22201/dgtic.30618096e.2025.3.4.135 Vol. 3,Núm. 4. octubre-diciembre 2025, págs. 62 - 64

Figura 25

OAI-PMH con el formato DIM



OAI Record: oai:ojs. <hostname> .:article/20

OAI Record Header

OAI Identifier	oai:ojs. <hostname> :article/20 **oai_dc** **formats**
Datestamp	2024-08-05T17:01:48Z
setSpec	dgtic:ART **Identifiers** **Records**

Unknown Metadata Format

```
<dim:dim xsi:schemaLocation="http://www.dspace.org/xmlns/dspace/dim http://www.dspace.org/schema/dim.xsd" >
    <dim:field mdschema="dc" element="title" >The prefix The title</dim:field>
    <dim:field mdschema="dc" element="description" > The abstract </dim:field>
    <dim:field mdschema="dc" element="contributor" qualifier="Author" email="submiter@mail.com" affiliation="" prefix="" >Submiter</dim:field>
    <dim:field mdschema="dc" element="contributor" qualifier="Diseñador" email="cristian.ortega@comunidad.unam.mx" affiliation="" prefix="" >Ortega
    Ramírez, Cristian Ricardo</dim:field>
    <dim:field mdschema="dc" element="publisher" ></dim:field>
    <dim:field mdschema="dc" element="date" qualifier="published" >2024-08-05</dim:field>
    <dim:field mdschema="dc" element="identifier" >article/20/publication/20</dim:field>
    <dim:field mdschema="rua" element="tipo_recurso" >62</dim:field>
    <dim:field mdschema="rua" element="descripcion_uso_educativo" >La descripción de uso e...</dim:field>
    <dim:field mdschema="rua" element="nivel_educativo" >3</dim:field>
    <dim:field mdschema="rua" element="categoria" >2</dim:field>
    <dim:field mdschema="rua" element="cita_apa" >Hernandez & Ortega (2024) ...</dim:field>
    <dim:field mdschema="rua" element="numero_isbn" >666</dim:field>
    <dim:field mdschema="rua" element="recurso_modificable" >2</dim:field>
    <dim:field mdschema="rua" element="tipo_proyecto" >4</dim:field>
    <dim:field mdschema="rua" element="clave_proyecto" >ACA2024</dim:field>
    <dim:field mdschema="rua" element="plantel_1" >F. Ciencias</dim:field>
    <dim:field mdschema="rua" element="nombre_plan_estudios_1" >Actuaria</dim:field>
    <dim:field mdschema="rua" element="anio_plan_1" >2020</dim:field>
    <dim:field mdschema="rua" element="semestre_anio_1" >10</dim:field>
    <dim:field mdschema="rua" element="nombre_asignatura_1" >Álgebra moderna</dim:field>
    <dim:field mdschema="rua" element="clave_asignatura_1" >2323</dim:field>
    <dim:field mdschema="rua" element="url_programa_1" >http://fciencia.unam.mx/algebra/plan.pdf</dim:field>
    <dim:field mdschema="rua" element="plantel_2" >p2</dim:field>
    <dim:field mdschema="rua" element="nombre_plan_estudios_2" >np2</dim:field>
    <dim:field mdschema="rua" element="anio_plan_2" >2021</dim:field>
    <dim:field mdschema="rua" element="semestre_anio_2" >1</dim:field>
    <dim:field mdschema="rua" element="nombre_asignatura_2" >na2</dim:field>
    <dim:field mdschema="rua" element="clave_asignatura_2" >ca2</dim:field>
    <dim:field mdschema="rua" element="url_programa_2" >url2</dim:field>
    <dim:field mdschema="rua" element="plantel_3" >p3</dim:field>
    <dim:field mdschema="rua" element="nombre_plan_estudios_3" >np3</dim:field>
    <dim:field mdschema="rua" element="anio_plan_3" >2022</dim:field>
    <dim:field mdschema="rua" element="semestre_anio_3" >8</dim:field>
    <dim:field mdschema="rua" element="nombre_asignatura_3" >na3</dim:field>
    <dim:field mdschema="rua" element="clave_asignatura_3" >ca3</dim:field>
    <dim:field mdschema="rua" element="url_programa_3" >url3</dim:field>
    <dim:field mdschema="rua" element="notas_plan" >mis notas</dim:field>
</dim:dim>
```

En la Figura 26, se muestran fragmentos del código fuente del *plugin* ruaOAIDim.

https://doi.org/10.22201/dgtic.30618096e.2025.3.4.135 Vol. 3,Núm. 4. octubre-diciembre 2025, págs. 63 - 64

Figura 26

Fragmentos del código fuente del plugin ruaOAIDim

```php
<?php
class DimMetadataFormat extends OAIMetadataFormat
{
    function __construct($prefix, $schema, $namespace)
    {
        parent::__construct($prefix, $schema, $namespace);
        PKPLocale::requireComponents([LOCALE_COMPONENT_PKP_SUBMISSION]);
    }

    function toXml($record, $format = null)
    {
        $article = $record->getData('article');
        $journal = $record->getData('journal');

        $templateMgr = TemplateManager::getManager();
        $templateMgr->assign(
            array(
                'journal' => $journal,
                'article' => $article,
                'issue' => $record->getData('issue'),
                'section' => $record->getData('section')
            )
        );

        // ...

        $ruaMetadata = [
            "nivel_educativo",
            "categoria",
            // ... [identificadores del resto de metadatos]
        ];

        $templateMgr->assign(
            array(
                // ... [metadatos de OJS]
                'abstract' => PKPString::html2text(
                  $article->getAbstract($article->getLocale())
                ),
                'ruaMetadata' => $ruaMetadata
            )
        );

        $plugin = PluginRegistry::getPlugin(
          'oaiMetadataFormats', 'DimMetadataFormatPlugin'
        );
        return $templateMgr->fetch(
          $plugin->getTemplateResource('record.tpl')
        );
    }
}
```

En la Figura 27, se muestran fragmentos de la plantilla que construye el xml en formato Dim.

https://doi.org/10.22201/dgtic.30618096e.2025.3.4.135 **Vol. 3,Núm. 4. octubre-diciembre** 2025, págs. 64 - 64

Figura 27

Fragmentos de la plantilla adaptada

```
<dim:dim xmlns:dim="http://www.dspace.org/xmlns/dspace/dim" xsi:schemaLocation="http://www.
dspace.org/xmlns/dspace/dim http://www.dspace.org/schema/dim.xsd">

{** ... metadatos de OJS ... **}
{if $article->getSource($journal->getPrimaryLocale()) }
    <dim:field mdschema="dc" element="source" >{$article->getSource
    ($journal->getPrimaryLocale())|escape}</dim:field>
{/if}
<dim:field mdschema="dc" element="identifier" >{$identifier|escape}</dim:field>

{** Metadatos RUA **}
{foreach from=$ruaMetadata item=metadata}
    {if $article->getData($metadata)}
        <dim:field mdschema="rua" element="{$metadata}" >{$article->getData($metadata)|
        escape}</dim:field>
    {/if}
{/foreach}

</dim:dim>
```