International Council on Archives
Conseil International des Archives

PAAG ICA
Photographic and Audiovisual Archives Working Group
Groupe Archives Photographiques et Audiovisuelles

ICA-PAAG. Guías breves

5

Título: Software para la gestión de la imagen

Autores: Rubén Alcaraz. EINA, Centre Universitari de Disseny i Art de Barcelona
Juan Alonso. Archivos Históricos de la Unión Europea

Association Loi 1901
SIRET N°784 226 128 000 14
APE 9900Z
Site web: www.ica.org

60 rue des Francs Bourgeois
75003 Paris, FRANCE
Tél : +33 (0)1 40 27 63 06
Fax : +33(0)1 42 72 20 65

ICA – Photographic and Audiovisual Archives Group (PAAG)
Dirección: David Iglésias i Franch

Autores: Rubén Alcaraz. EINA, Centre Universitari de Disseny i Art de Barcelona
Juan Alonso. Archivos Históricos de la Unión Europea

Publicación: Noviembre 2017

Software para la gestión de la imagen

0. Introducción

La fotografía digital en los archivos ha generado un escenario marcado por un gran volumen de imágenes, escenario que debe considerar aspectos como resolución, formatos, modos y perfiles de color, compresiones, sistemas de capturas, metadatos, dispositivos de salida y nuevos soportes físicos, entre otros. En esta circunstancia, una de las necesidades más críticas es dotarse del software de gestión para llevar a cabo las tareas asociadas a la creación y al mantenimiento de un archivo digital.

En la última década y, en la línea de lo sucedido en otros ámbitos como en el de los sistemas de gestión de contenidos (CMS), el mercado de las aplicaciones específicas para la gestión y difusión de fotografías ha crecido exponencialmente, configurando un sector complejo y sumamente competitivo. Por un lado, se advierte una gran diversidad respecto a la especialización de los productos disponibles, ya que existe una gran cantidad de aplicaciones focalizadas en tareas concretas. Esta especialización implica que, en muchos casos, se haga necesaria la implementación de diferentes soluciones que deben coexistir e interoperar en vistas a resolver todos o varios de los requisitos del archivo digital. Por otro lado, además de esta diversidad funcional de soluciones, en el mercado también encontramos programas orientados a diferentes ámbitos, como son el editorial, bibliotecario, archivístico o museístico. Finalmente, ante la dificultad inherente para seleccionar una solución entre la inmensa cantidad de aplicaciones disponibles, no existe un consenso generalizado en cuanto a nomenclatura se refiere, y así pueden encontrarse en la bibliografía referencias a un mismo software bajo nombres dispares.

Esta guía tiene como propósito ofrecer al lector una panorámica de las principales tipologías de software para la gestión de imágenes digitales y, a su vez, proporcionar una serie de criterios básicos para la selección de aplicaciones.

1.Tipos de software

Las categorías propuestas no son excluyentes, sino que representan diferentes aspectos del objeto de estudio. De esta manera, los programas con licencia de software libre y los privativos pueden presentarse tanto en forma de instalación local, como en la nube, y distribuirse gratuitamente, o bajo cualquier otra de las modalidades que destacamos a continuación.

1.1. Según el tipo de licencia

1.1.1. Libre

Cuando hablamos de software libre, nos referimos a una cuestión de libertad, no de precio. Es importante destacar esta diferencia por la ambigüedad inherente a la polisemia de la voz inglesa *free*. En concreto, el software libre se asocia a la libertad para ejecutar, copiar, distribuir, estudiar, modificar y mejorar el software. De acuerdo a Stallman (2004), las libertades que definen al software como libre son:

- Libertad 0: libertad para ejecutar el programa sea cual sea nuestro propósito.
- Libertad 1: libertad para estudiar el funcionamiento del programa y adaptarlo a nuestras necesidades (el acceso al código fuente es indispensable).
- Libertad 2: libertad para redistribuir copias del programa.
- Libertad 3: libertad para mejorar el programa y posteriormente distribuirlo para el bien de toda la comunidad.

Las licencias libres por excelencia son las conocidas como GNU GPL (GNU General Public License), aunque la gran cantidad de licencias de software libre existentes obliga a prestar atención a sus cláusulas para determinar el alcance de las libertades que nos confieren.

Una de las principales ventajas relacionadas con el uso del software libre tiene que ver con el ahorro de costes en relación a las licencias de uso, pero sus ventajas no solo se limitan al aspecto económico. La posibilidad de adaptar las aplicaciones a nuestras necesidades está totalmente garantizada gracias al acceso al código fuente. Por otro lado, la supervivencia de la aplicación no depende exclusivamente de su rentabilidad o de consideraciones estrictamente económicas, sino del interés de la comunidad de usuarios y desarrolladores. El libre acceso al código fuente es una garantía de durabilidad e independencia tecnológica, aún si la aplicación ya ha dejado de tener el soporte mayoritario de la comunidad. Al software libre también se le asocian otras ventajas, como la posibilidad de escoger a cualquier empresa para que se encargue de la implementación, desarrollo y mantenimiento de nuestro sistema (no solo a la empresa desarrolladora oficial), o la facilidad para migrar o establecer conexiones con nuevos entornos o sistemas, ya que la mayoría de las aplicaciones de software libre trabajan con estándares y formatos abiertos.

1.1.2. Código abierto

La principal diferencia entre las licencias libres y las licencias de código abierto es que estas últimas, a pesar de permitir el acceso y modificación del código fuente, generalmente no permiten redistribuir las modificaciones o hacer un uso comercial de las mismas, por lo que limitan algunas de las libertades garantizadas por las primeras. Más allá de las diferencias filosóficas, las licencias de código abierto se centran en las ventajas propias de un modelo de desarrollo que, como en el caso del software libre, se caracteriza por una dinámica horizontal en la que una comunidad con acceso al código fuente colabora aportando sus conocimientos en diferentes áreas (programación, diseño de la interfaz, localización, etc.), y que dará como resultado un producto más robusto y de mayor calidad.

1.1.3. Privativo / propietario

El llamado software privativo (o propietario) es aquel que no ofrece una forma libre de acceso a su código fuente, el cual solo se encuentra a disposición de su desarrollador y no permite su libre modificación ni lectura por parte de terceros. A pesar de que, como en el caso del software libre, hablamos de libertad y no de coste de producto, buena parte del software privativo es de distribución comercial. Como en el caso del software libre y del de código abierto, las condiciones de uso se encuentran especificadas en un contrato (licencia) entre el autor o distribuidor del software y el licenciatario (usuario) en el que se establecen los términos y condiciones por los que se regirá la relación entre ambos.

1.2. Según el tipo de instalación

1.2.1. Local (*on premises*)

Cuando se habla de software local (u *on premises*) se hace referencia a aquel software instalado en los servidores propiedad de la institución (modelo tradicional). Las ventajas de este tipo de instalación son la posibilidad de un mayor control de las actualizaciones y configuraciones técnicas del aplicativo, una supuesta mejor transferencia de archivos (ya que se realiza a través de una red local) y una mayor seguridad. Por otro lado, exige una mayor responsabilidad en el mantenimiento y la evolución del aplicativo puede ser más lenta.

1.2.2. Nube (*cloud*)

La computación en la nube es un tipo de computación que se fundamenta en el acceso a una serie de recursos compartidos (hardware y software) a través de Internet, generalmente en forma de servicios. Los usuarios y administradores pueden usar la computación en la nube sin necesidad de disponer grandes conocimientos técnicos sobre su funcionamiento interno pero, a su vez, sin tener el control de las tecnologías que se hayan detrás de estos servicios.

En la mayoría de los casos, los clientes de la computación en la nube no son los propietarios de la infraestructura, sino que pagan por el “alquiler” de la misma. La posibilidad de virtualizar entornos añade a la computación en la nube una gran escalabilidad, que se traduce en la

posibilidad de ampliar o reducir bajo demanda las características de los servicios contratados. Otras de sus ventajas son la posibilidad de contar con una infraestructura de última generación "inmune" a la rápida evolución del sector tecnológico sin necesidad de realizar una fuerte inversión inicial y la posibilidad de contar con soporte 24/7 para solventar errores que puedan afectar a la infraestructura. Entre las desventajas, destacan una mayor dependencia del proveedor de servicios, la necesidad de disponer de conexión a Internet de manera permanente, una menor velocidad en el acceso a los datos frente a las infraestructuras en local y una cierta vulnerabilidad en la seguridad de nuestros datos, que será más o menos pronunciada según las garantías que ofrezca cada proveedor.

1.3. Según el tipo de distribución

1.3.1. *Freeware*

El término *freeware*, resultado de la contracción de las palabras *free* (en su acepción de gratuito) y la terminación de *software*, se refiere a aquellas aplicaciones gratuitas cuyas licencias permiten la redistribución del programa, pero no su modificación, pues su código fuente normalmente no se encuentra disponible. Las aplicaciones de este tipo no son, por lo tanto, libres, sino privativas. En algunos casos, bajo esta categoría encontramos versiones reducidas (*lite*) de aplicaciones de pago profesionales, o versiones de esa misma aplicación que contienen diferentes tipos de publicidad, según el caso.

1.3.2. Comercial

El software comercial es aquel desarrollado por una empresa como parte de su actividad comercial. Las compañías desarrolladoras cobran por el producto, distribución y/o soporte.Cabe destacar que no todo el software comercial es privativo. A pesar de que la gran mayoría del programario comercial lo es, las licencias de software libre no son incompatibles con su comercialización ni con la prestación de servicios asociados. En este sentido, existe software libre que es comercial (se distribuye respetando las 4 libertades mencionadas anteriormente) y software privativo que no es comercial (*freeware* o gratuito).

1.3.3. Versiones de prueba (*shareware*)

El término *shareware* se refiere a aquellas aplicaciones cuya licencia permite la redistribución de copias de un ejecutable que generalmente se caracteriza por incluir algunas limitaciones o restricciones respecto a las características finales del producto. El acceso a la versión completa está condicionado al pago y aceptación de una licencia privativa. El software *shareware* no puede ser considerado libre porque su código fuente no se encuentra disponible, por lo que no puede ser estudiado, ni modificado. También es conocido como *trialware*, *demoware*, *freemium*, etc.

1.3.4. SaaS (*Software as a Service*)

Aunque en ocasiones puede existir una cierta confusión entre los conceptos de *cloud* y *SAAS* (*Software As A Service*), la realidad es que esta última se refiere a un modelo de negocio a través del cual un proveedor determinado ofrece acceso a una o diversas instancias de una

aplicación alojadas en su entorno, a las cuales el cliente tiene acceso a través de Internet. La forma de contratación del servicio generalmente reviste la forma de suscripción de carácter mensual o anual. De esta manera, como en el caso de los modelos *cloud*, las aplicaciones no residen en los servidores u ordenadores del cliente, sino que se encuentran disponibles en la nube, mientras se satisfaga la cuota correspondiente. Una vez dejemos de pagar la suscripción, dejaremos de tener acceso a la aplicación y a los datos contenidos en ella.
La mayor parte de empresas que ofrecen servicios SAAS complementan el acceso al software con un servicio de mantenimiento y soporte, de manera que el cliente solo debe preocuparse del uso de la aplicación y no de su administración técnica.

1.4. Según el tipo de funcionalidad

A continuación, se listan y definen 6 clases de aplicaciones que se consideran útiles en la gestión de un archivo de imágenes. De cada tipo se nombran algunos ejemplos cuyas principales funcionalidades pueden ser consultadas en el siguiente recurso: https://www.ica.org/en/survival-kit-software-0

1.4.1. Extractores de metadatos

Se trata de herramientas que permiten la lectura y extracción de metadatos técnicos (captura y edición) y descriptivos. Son útiles para poder disponer de más información acerca del contexto de la imagen (identificación de autor, coordenadas GPS, cámara utilizada, modelo de lente, apertura, flash, fecha, entre otros) y/o realizar una extracción de metadatos para ser explotados por otro sistema. Algunos ejemplos son Exiftool, NLZN Metadata Extractor Tool, PhotoMe o AsTiffTagViewer.

1.4.2. Visores

Son aplicaciones que permiten visualizar y gestionar imágenes digitales en diferentes formatos. Están especialmente pensadas para una visualización rápida de las imágenes, aunque también ofrecen interesantes funcionalidades en la ingesta, en el tratamiento de metadatos, en el procesamiento básico y en la salida. Existe una gran oferta de soluciones que, en ocasiones, hace difícil escoger una solución única y/o principal. No obstante, no son aplicaciones restrictivas, de manera que, por ejemplo, es posible emplear un programa para la importación, otro para la incrustación de metadatos en lote y otro para la selección de las imágenes de acuerdo a las políticas de valoración establecidas por la institución. Algunos ejemplos son IrfanView, FastStone Image Viewer, XnView o Adobe Bridge.

1.4.3. Editores

Los editores son programas especialmente diseñados para crear y editar imágenes de forma interactiva y guardarlas en un formato de archivo gráfico como JPEG, PNG, GIF o TIFF, entre otros. Ofrecen mayores y más avanzadas posibilidades de procesamiento que los visores (trabajo con capas, edición por zonas y gestión de espacios de color, entre otras). Podríamos distinguir un subtipo de editor enfocado a la edición masiva; es decir, capaz de editar varias imágenes de forma simultánea. Algunos ejemplos son Adobe Photoshop, Gimp, Adobe Lightroom o Darktable.

1.4.4. Software de identificación y validación de formatos

Las aplicaciones de análisis de formatos permiten la identificación y validación de un determinado formato. Con ello, se obtiene la verificación del formato que habitualmente anuncia la extensión del archivo y se comprueba el grado de correspondencia con la estructura que corresponde. Esta operación es especialmente útil en el momento de la ingesta. Algunos ejemplos son Droid, Jhove2, Jpylyzer o DPF Manager.

1.4.5. Sistemas de gestión de contenidos (CMS)

En la última década, los sistemas de gestión de contenidos (CMS) han popularizado la creación de sitios web entre usuarios que no disponen de demasiados conocimientos técnicos en el desarrollo de este tipo de productos. En el ámbito de la información y la documentación, se ha venido observando una convergencia entre las tradicionales aplicaciones de escritorio para la gestión de los fondos de las unidades de información y los CMS, lo que ha dado lugar al surgimiento de aplicaciones web específicas que integran tanto las características tradicionales del software documental como opciones de publicación de esos contenidos en la Web. Algunos ejemplos son ArchivesSpace, AtoM, CollectiveAccess, u Omeka.

1.4.6. *Digital Asset Management* (DAM)

Las soluciones DAM aparecen en el mercado a finales de los 90 para mejorar la gestión de los activos digitales en el sector del *gaming* y de los medios de comunicación, pero han ido expandiéndose a otros sectores, como el archivístico. Estos sistemas tienen una voluntad integral y centralizan y agilizan la ingesta y el tratamiento documental de contenidos digitales, así como el control y seguimiento de su uso, los procesos asociados con la propiedad intelectual y los derechos, la seguridad y su búsqueda y recuperación. La diferencia con los CMS del sector documental puede ser sutil, ya que estos han mejorado mucho sus funcionalidades. Generalizando, los DAM están más optimizados en los flujos de trabajo, las funciones de búsqueda y de intercambio de documentos, mientras que no están tan pensados como los anteriores para la difusión en línea. No obstante, muchos de ellos ofrecen una buena salida web o permiten integraciones con CMS. Hay diferentes tipos de DAMs, pero básicamente podemos diferenciar entre sistemas pensados para la gestión de archivos definitivos y sistemas más enfocados en la producción que incluyen funcionalidades relacionadas con la edición, aprobación y revisión de versiones, entre otros. Algunos ejemplos son Celum, Fotoware, Picturepark o Razuna.

2. Criterios de selección de software de carácter integral

2.1. Análisis de la situación de partida y detección de necesidades

Aspectos como la gran oferta disponible en el mercado, la complejidad de estas aplicaciones, la necesidad de someterlas a diferentes pruebas o el mismo hecho de que una mala elección pueda condicionar el proyecto para el que se desean poner en marcha, hacen del proceso de selección la fase más importante dentro de un proyecto de implementación de un software como el que nos ocupa.

La situación de partida y las particularidades de cada institución marcarán una serie de condicionantes que deberán ser incorporados como requisitos al proyecto. Entre estos aspectos encontramos factores relacionados con la capacidad económica de la institución que, además, deberán contemplarse a largo plazo para asegurar la sostenibilidad de la solución implementada. También encontramos otros condicionantes, como la existencia de una aplicación anterior desde la cual debemos migrar la información al nuevo sistema, la infraestructura tecnológica disponible, la necesidad de cumplir con algún estándar o requisito legal o la necesidad de ser interoperables con terceras plataformas o servicios, entre otros.

2.2. Concreción, categorización y priorización de requisitos

Una vez realizado el análisis inicial, las necesidades detectadas deberán ser expresadas en forma de requisitos concretos para, posteriormente, categorizarlos y priorizarlos. Los requisitos de un sistema describen de forma precisa las propiedades que debe satisfacer el software y las restricciones asociadas a su funcionamiento.

Existen diferentes metodologías pensadas para abordar la selección de una aplicación como la que nos ocupa. Entre estas, destacamos el método MosCoW, una técnica utilizada en gestión de proyectos, análisis de negocios o en el desarrollo y selección de aplicaciones de software. El término MoSCoW es un acrónimo derivado de la primera letra de cada una de las categorías propuestas en vistas a identificar la prioridad de las diferentes funcionalidades requeridas. Estas son:

- Must have: funcionalidad o característica imprescindible de la cual no se puede prescindir y que, por lo tanto, debe ser cubierta por la aplicación desde su implantación.
- Should have: funcionalidad o característica necesaria que puede ser cubierta manualmente de manera temporal y que, por tanto, no es imprescindible en la fecha de lanzamiento. No obstante, se debe planificar su implementación en un plazo razonable de tiempo, en vistas a sustituir la solución temporal.
- Could have: funcionalidad o característica que no es estrictamente necesaria pero que, por el hecho de tenerla, por un pequeño coste adicional incrementará la satisfacción y/o productividad de los miembros de la organización. Bajo ninguna circunstancia debería comprometer la fecha de entrega.
- Would like to have: Funcionalidades menos críticas, innecesarias en el desarrollo actual, pero que estaría bien tener en el futuro como parte de un proyecto de mejora.

Una vez recopilados todos las requisitos planteados y organizados en las diferentes categorías propuestas por el método MoSCoW, obtenemos una escala nominal en la que los requisitos asociados a cada una de las categorías representan la misma prioridad. Si se considera necesario, los requisitos asociados a cada categoría pueden ser ponderados de acuerdo a su nivel de cumplimiento. En este sentido, pueden definirse, entre otros, los siguientes:

- Core: funcionalidad o componente inherente incorporado al núcleo de la aplicación en la versión actual de la aplicación.

- Third party: funcionalidad no inherente al núcleo de la aplicación, pero que se puede integrar en el sistema mediante módulos existentes ofrecidos por terceros.
- Desarrollo: funcionalidad no inherente al núcleo de la aplicación, pero que puede ser desarrollada por la empresa o el departamento de informática encargados de su implementación.
- Planificado: funcionalidad no disponible en la versión actual de la aplicación, pero que se prevé sea incorporada en siguientes versiones dentro del núcleo o como módulo.
- No cumplida: funcionalidad que la aplicación no cumple a partir de su núcleo, ni con ningún módulo o extensión disponible. Tampoco se prevé su desarrollo a corto o medio plazo.

Tipo de requisito	**Nivel de cumplimiento**	**Puntuación**
Must have (x4)	Core (x6) Third party (x3) Desarrollo (x2) Planificado (x1)	24 puntos 12 puntos 8 puntos 4 puntos
Should have (x3)	Core (x6) Third party (x3) Desarrollo (x2) Planificado (x1) No cumplido (x0)	18 puntos 9 puntos 6 puntos 3 puntos 0 puntos
Could have (x2)	Core (x6) Third party (x3) Desarrollo (x2) Planificado (x1) No cumplido (x0)	12 puntos 6 puntos 4 puntos 2 puntos 0 puntos
Would like to have (x1)	Core (x6) Third party (x3) Desarrollo (x2) Planificado (x1) No cumplido (x0)	6 puntos 3 puntos 2 puntos 1 puntos 0 puntos

El vencedor de la comparativa será aquel sistema que resuelva una mayor cantidad de requisitos, teniendo en cuenta que el orden de funcionalidades de acuerdo a su importancia es: must have, should have, could have y would like to have y que, por lo tanto, las más prioritarias deben tener un mayor peso sobre las que tienen un nivel de prioridad menor.

A la hora de establecer la lista de requisitos, debemos confeccionar una lista numerada que, en la medida de lo posible, sea totalmente exhaustiva; es decir, que tenga en cuenta todas las funcionalidades necesarias atendiendo a las características de cada centro, las necesidades de

los diferentes usuarios y los tipos de objetos digitales con los que se va a trabajar, así como los procesos internos que queremos automatizar, entre otros aspectos comentados anteriormente. Asimismo, cada requisito deberá ser:

- relevante: evitará formular requisitos que no se atengan a las necesidades reales.
- específico: deberá expresar claramente cuál es la funcionalidad o característica que debe satisfacer.
- preciso: no podrá ser descompuesto en diversas funcionalidades a su vez.
- verificable: su cumplimiento deberá poder ser evaluado en una prueba.

A continuación se muestra una serie de ejemplos de requisitos, la funcionalidad a la que se refiere desarrollada, y el nivel de prioridad.

	Requisito	**Funcionalidad**	**Prioridad**
1	Dublin Core	Soporte para la asignación y gestión del conjunto de elementos de metadatos Dublin Core.	Must have
2	Identificación de duplicados	Identificación de duplicados a través del nombre de fichero y a través del metadato UID (Unique Image ID) del estándar EXIF-TIF.	Should have
3	Formularios dinámicos	Formularios dinámicos de manera que los campos de descripción varíen en función de los campos seleccionados.	Should have
4	Análiticas	Extracción de analíticas de descargas de imágenes.	Could have
5	Visualización de vídeos	Videostreaming de los ficheros de vídeo en formato MOV, AVI, MPEG2, MPEG-4 y MXF	Could have

Entre las categorías de requisitos más frecuentes asociadas a las aplicaciones que nos ocupan encontramos:

- Ingesta de contenidos: dentro de la que se incluyen todas las funcionalidades asociadas a la carga de ficheros en la aplicación, importación por lotes, ingesta externa, plantillas de ingesta, extracción automática de metadatos, etc.
- Metadatos: soporte para la descripción de las imágenes mediante los conjuntos de elementos de un determinado esquema de metadatos, importación y exportación de metadatos en diferentes formatos (RDF, XML, etc.).
- Herramientas de administración y de flujo de trabajo: todo tipo de funcionalidades asociadas al trabajo diario del documentalista como búsquedas avanzadas, edición en

lote de registros, gestión de solicitudes de usuarios, herramientas de comunicación, posibilidades de edición de imágenes, tipos de descarga o envío, etc.

- Seguridad, gestión de usuarios y permisos: funcionalidades asociadas a la gestión de diferentes perfiles de usuario con distintos niveles de acceso a la aplicación, así como características que aseguren la seguridad de la aplicación, especialmente si ofrecemos acceso a ella desde Internet.
- Interfaz de consulta: incluye todas aquellas características que permiten, tanto a los administradores, como a los usuarios, realizar consultas sobre la base de datos, limitar o ampliar resultados, guardar favoritos, ver documentos relacionados, acceder desde dispositivos móviles, etc.
- Soporte y documentación: bien a través del desarrollador oficial o, en el caso de las aplicaciones de software libre, a través de la disponibilidad de empresas de servicios relacionadas, comunidades de usuarios, manuales publicados, etc.

2.3. Búsqueda y preselección de soluciones

Actualmente, es relativamente fácil encontrar suficiente información en la Red como para realizar una preselección de soluciones a analizar de manera informada. Más allá del uso de buscadores generalistas, a los que podemos interrogar con palabras clave como “dam software” o “Digital Asset Management Software”, en Internet podemos encontrar diferentes recursos de interés como:

- Directorios de aplicaciones: además de ofrecer acceso a los sitios web oficiales de cada aplicación, pueden contener información básica sobre estas, así como recursos de valor añadido, como demos o la valoración de los usuarios
- Blogs especializados: como pasa en otros ámbitos, el interés que despiertan aplicaciones como los CMS o DAM se ha traducido en una importante cantidad de medios digitales dedicados al análisis y la publicación de noticias de actualidad sobre este ámbito. Los blogs son recursos de interés tanto para descubrir aplicaciones nuevas como para mantenernos informados acerca de las novedades de las ya consolidadas. Con respecto a este tipo de recursos, es importante saber detectar contenido patrocinado.
- Artículos científicos: son varias las revistas científicas que dedican alguna de sus secciones a la reseña de software. Adicionalmente, también es posible encontrar artículos que explican las experiencias de una institución en la implementación o migración de aplicaciones informáticas.
- Empresas de consultoría: bien a través de sus sitios web, bien a través de blogs, las empresas de consultoría que ofrecen servicios en torno a las aplicaciones que nos ocupan pueden ser otra importante fuente de información acerca del mercado relacionado con la fotografía digital. Igualmente, existen empresas especializadas que ofrecen servicios de acompañamiento en la búsqueda de la solución más adecuada. La visión de un profesional externo puede ayudar también en la elaboración de una lista de requisitos más completa, mientras que la experiencia del mercado permitirá un análisis

más crítico del cumplimiento de los requisitos de las soluciones candidatas. Es importante que estas empresas consultoras manifiesten su carácter independiente, para asegurarnos que no estén vinculadas a un software concreto.

2.4. Comparativa

Una vez tengamos nuestra lista de requisitos y nuestra lista de soluciones candidatas, es momento de contactar con las empresas desarrolladoras o sus proveedores para valorar la aplicación en profundidad. El grado de transparencia informativa de las soluciones es diverso y, en ocasiones, las prestaciones son demasiado genéricas. La posibilidad de reunirse con los comerciales y técnicos del software mediante demostraciones generales, o incluso solicitando una versión de prueba del aplicativo, ayudará a esclarecer si se cumplen o no algunos requisitos y, en el caso de que se cumplan, ver de qué manera lo hacen. La lista finalista de candidatos no debería ser de más de 4 aplicaciones. Una última criba puede realizarse en la propia institución con una demostración *in situ* que incluya material y procesos propios de la institución y la resolución de una serie de requisitos críticos.

2.5. Consejos finales

- Como hemos comentado anteriormente, la experiencia de otras instituciones, especialmente si son similares a la nuestra, puede ser de gran ayuda en la decisión final. En este sentido, es interesante solicitar al proveedor o desarrollador una lista de clientes a los que se les pueda interrogar o visitar de forma privada. No obstante, seleccionar una aplicación simplemente porque una institución de referencia la utiliza puede ser un gran error. El tamaño de la institución, sus objetivos y la situación de partida son, entre otros, aspectos que debemos tener en cuenta.
- El conocimiento o experiencia previa del departamento de informática o del documentalista con un software determinado puede ser un punto a favor en la selección, pero la decisión final nunca debe basarse exclusivamente en ese tipo de criterios.
- En el proceso de análisis, evaluación y decisión final deben participar todos los actores involucrados: documentalista, informáticos, alta dirección, etc.
- Si optamos por una solución de software libre, conviene asegurarnos de la existencia de una comunidad de usuarios activa, así como de empresas dedicadas a dar soporte y servicios relacionados.
- Aunque en el caso de las aplicaciones privativas puede ser complicado de hallar, es conveniente revisar la hoja de ruta de desarrollo de la aplicación, así como el historial de versiones publicado hasta el momento. En este sentido, es interesante optar por aplicaciones que se actualizan con frecuencia, corrigiendo errores o añadiendo nuevas funcionalidades, asegurando un soporte a largo plazo.
- En la medida de lo posible, debemos intentar optar siempre por aplicaciones que trabajen con estándares; si son abiertos, mucho mejor.

Bibliografía

Abbot, Leala. *The DAM list: tech specs* <https://docs.google.com/spreadsheets/u/1/d/1xRwkQVluqtlLVeuLgHtx3EtZeNAye3n_7BwR13GKwm0/pub?hl=en_US&hl=en_US&single=true&gid=0&output=html>. [Consulta: 09/10/2017].

Alcaraz Martínez, Rubén. " El CollectiveAccess, un sistema de gestió i difusió de col·leccions de museus, arxius i biblioteques ". *BiD: textos universitaris de biblioteconomia i documentació*. Núm. 33 (des. 2014). <http://bid.ub.edu/es/33/alcaraz2.htm>. [Consulta: 09/10/2017].

Alcaraz Martínez, Rubén. "Omeka: exposicions virtuals i distribució de col·leccions digitals". *BiD: textos universitaris de biblioteconomia i documentació*, Núm. 28 (juny 2012). <http://bid.ub.edu/28/alcaraz2.htm>. [Consulta: 09/10/2017].

CMS critic. <https://www.cmscritic.com>. [Consulta: 09/10/2017].

Dam maturity model. 2017. <http://dammaturitymodel.org/>. [Consulta: 09/10/2017].

DAM news. London: Daydream. <http://digitalassetmanagementnews.org/>. [Consulta: 09/10/2017].

Diamond, David. *DAM survival guide: Digital Asset Management initiative planning.* [Seattle]: CreateSpace, 2012.

Frey, Franziska; Williams-Allen, Shellee; Vogl, Howard; Chandra, Levey. *Digital Asset Management: a closer look at the literature.* Rochester: Rochester Institute of Technology, 2005. <http://scholarworks.rit.edu/books/3/>. [Consulta: 09/10/2017].

G2 Crowd. <https://www.g2crowd.com/categories/digital-asset-management>. [Consulta: 09/10/2017].

Government of Canada. ''Digital Asset Management and museums: an introduction''. Última fecha de actualización: 2017-08-27. <http://canada.pch.gc.ca/eng/1442946637162>. [Consulta: 09/10/2017].

McGathMad, Gary. *File format science*. 2015. <https://madfileformatscience.garymcgath.com> [Consulta: 09/10/2017].

Open Preservation. Wetherby, West Yorkshire: Foundation Open Preservation Foundation. <http://openpreservation.org/>. [Consulta: 09/10/2017].

SCAPE Scalable Preservation Environments. Seibersdorf : AIT Austrian Institute of Technology. <http://scape-project.eu/>. [Consulta: 09/10/2017].

Open Source Digital Asset Management <http://www.opensourcedigitalassetmanagement.org>. [Consulta: 09/10/2017].

Stallman, Richard M. *Software libre para una sociedad libre*. Madrid: Traficantes de Sueños, 2004. <http://www.gnu.org/philosophy/fsfs/free_software.es.pdf>. [Consulta: 03/06/2017].

The Institute of Electrical and Electronics Enginee. *IEEE Recommended practice for software requirements specifications: IEEE Std 830-1998*. New York: IEEE, 1998.