Chapitre 6

Métadonnées et document

Sommaire

Dans le chapitre précédent, nous avons présenté le modèle de document utilisé dans le framework Sydonie, basé sur le cadre conceptuel des FRBR. En fonction de la nature des données, notre modèle de document permet de placer des informations à divers niveaux d'abstraction grâce aux entités Work, Expression et Manifestation.

Dans ce chapitre, nous présentons les problématiques liées à la gestion des métadonnées et comment le framework Sydonie tire parti du modèle défini précédemment. Cette gestion à différents niveaux va se révéler utile pour la gestion et la qualité des métadonnées stockées pour les documents.

6.1 Un framework pour les métadonnées

Nous avons décrit dans la section 3.3 les problématiques auxquelles fait face un développeur d'applications lorsqu'il doit gérer les métadonnées des documents. Nous en avons déduit les points de blocage et préconisé les apports qu'un framework doit fournir pour simplifier le travail du développeur, que nous résumons ici :

– gestion de l'extraction des métadonnées lors de l'ajout de fichiers dans le système ;

– gestion de l'inclusion des métadonnées dans les fichiers gérés par le système. Cette opération doit permettre la cohérence des données présentes dans le système et dans le fichier ;
– gestion des correspondances entre les données du modèle de l'application et le(s) schéma(s) de métadonnées utilisé(s).

Ce chapitre présente les solutions apportées par Sydonie pour répondre aux besoins exprimés ci-dessus et la façon dont les développeurs peuvent tirer profit de l'utilisation du modèle de document de Sydonie pour la gestion des métadonnées.

Au fil de ce chapitre, nous utiliserons un exemple concret pour illustrer les processus en jeu et les solutions mises en place. L'application `lisez-moi.lu`, en cours de développement par C&Féditions, est une plate-forme de vente de livres numériques, à destination des petits éditeurs. Dans le cadre de cette plate-forme, un éditeur gère les livres dont il souhaite la vente en ligne. Un livre numérique peut être disponible sous divers formats comme ePub ou PDF par exemple. S'il est important de noter que les divers formats de fichiers ne sont pas générés par un processus automatique, on considère ici qu'un nouveau type de fichier correspond à une nouvelle Manifestation. En effet, on peut faire le parallèle avec une le cas d'édition différente puisque le contenu intellectuel est le même. Seuls changent l'apparence et le contenant.

Dans le cadre de `lisez-moi.lu`, un type de document eBook a été créé au sein de l'application, en utilisant le modèle présenté au chapitre précédent. Avec ce modèle, un objet de type eBook aurait donc les propriétés :

– au niveau de l'entité Work : titre uniforme, date de première publication, auteur (un ou plusieurs). Rappelons que ces propriétés sont valables pour l'ensemble de l'arborescence du document ;
– au niveau de l'entité Expression : titre, description, langue, auxquels on ajoute l'attribut éditeur. Chaque Expression correspond à une version linguistique différente de l'œuvre ;
– au niveau de l'entité Manifestation : type de fichier, taille du fichier, auxquels on ajoute les attributs prix, ISBN, copyright. Ces propriétés dépendent du fichier lui-même.

Notons que, comme nous l'avons recommandé dans la partie 3.3.7, le modèle de l'application est alors créé de façon complètement indépendante des métadonnées que celle-ci devra intégrer dans les fichiers ou pages web.

6.2 Processus de gestion des métadonnées

6.2.1 Exemple

Dans le cas de `lisez-moi.lu`, l'ajout d'un livre au catalogue commence par l'upload d'un fichier, par exemple au format ePub. Si le fichier ePub a été créé de façon professionnelle, le fichier doit d'ores et déjà contenir un certain nombre de métadonnées concernant le livre. En

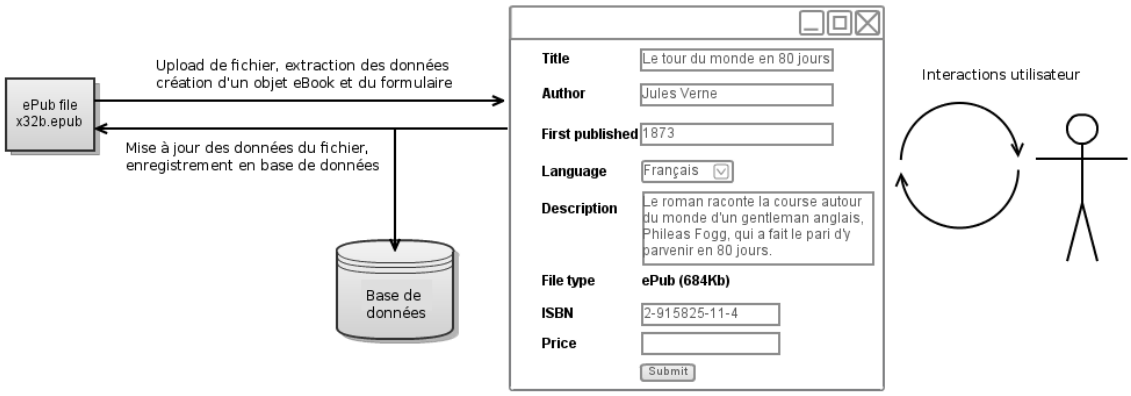

FIGURE 6.1 – Processus d'ajout d'un livre au catalogue

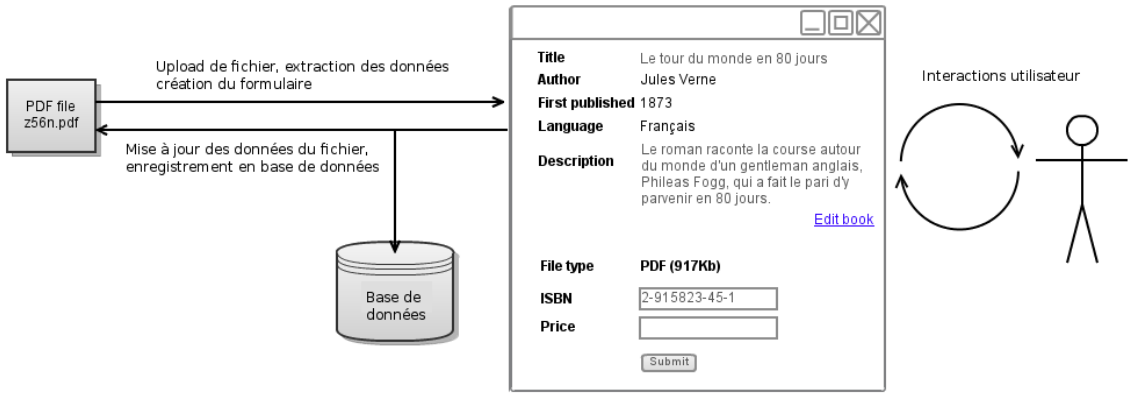

FIGURE 6.2 – Processus d'ajout d'un fichier pour un livre existant

effet, un fichier ePub définit son contenu à l'aide de données exprimées en XML au format OPF (Open Packaging Format) [OPF 07]. Ces données décrivent principalement les composants du fichier ePub et un ensemble de métadonnées concernant le livre numérique. Dans la norme ePub version 2, les métadonnées sont exprimées à l'aide des éléments du Dublin Core.

Lors de l'ajout d'un fichier, il serait donc aberrant de ne pas proposer à l'éditeur d'utiliser ces informations, charge à lui de les vérifier. L'éditeur se voit alors proposer un formulaire pré-rempli avec les informations que le système a pu extraire du fichier. L'éditeur peut alors vérifier et compléter les informations concernant le livre. Une fois ces informations validées, les différentes données sont enregistrées dans le système en utilisant le modèle de document défini. Enfin, pour assurer la cohérence entre les données de l'application et les métadonnées du fichier, les métadonnées du fichier peuvent être réinscrites dans celui-ci. Le processus décrit est illustré figure 6.1.

Lors de l'ajout, pour le même livre, d'un autre type de fichier, par exemple au format PDF, le processus sera quelque peu différent. En effet, pour l'application il ne s'agit pas ici de créer un

nouvel objet eBook, mais il s'agit d'ajouter une Manifestation à un objet eBook existant. Cette distinction doit bien évidemment être transparente pour l'éditeur. Dans ce cas, les informations *déjà présentes* dans le système sont utilisées. Les données extraites du fichier déposé servent à pré-remplir un formulaire qui ne concerne que les informations spécifiques au fichier PDF ajouté pour le livre existant. Là encore, une fois les informations validées par l'éditeur, les données sont mises à jour dans le fichier pour assurer la cohérence globale. Ce processus est illustré figure 6.2.

6.2.2 Généralisation du processus

Le processus présenté dans la section précédente est généralisé pour la gestion des fichiers qu'un utilisateur ajoute au système. En effet, Sydonie a vocation à gérer des documents, en particulier nous avons souligné les manques des systèmes existants quant à la gestion des fichiers déposés, le cas des images illustrant un contenu étant le plus courant. L'ajout de fichier dans une application gérée par Sydonie passe par l'étape d'extraction des métadonnées présentes dans le fichier. Le développeur de l'application est alors responsable de ce qu'il advient de ces données et de la façon dont elles sont intégrées ou non au système. Afin de faciliter ces opérations, Sydonie réalise les principales opérations, que nous allons détailler dans les parties suivantes.

6.3 Conteneur de métadonnées

Malgré l'utilisation possible de XMP pour intégrer des métadonnées dans les fichiers, la gestion des multiples formats doit être assurée afin de garantir la cohérence des métadonnées. Par exemple, dans une image les métadonnées seront exprimées en utilisant différents schémas : Exif, IPTC et/ou XMP. De plus, comme l'exemple de `lisez-moi.lu` le montre, des métadonnées identiques peuvent être insérées dans des types de fichiers différents. Enfin, les métadonnées doivent pouvoir être lues (extraites) et écrites (insérées) dans les fichiers.

Les métadonnées pour un document peuvent être générées de deux façons : à partir du modèle de l'application (création d'un objet à partir de rien), ou bien à partir de fichiers (cas de l'import d'un fichier pour créer un document par exemple). Le framework Sydonie utilise un conteneur de métadonnées comme pivot entre le modèle d'application et les métadonnées intégrées dans les fichiers. Le conteneur stocke les informations de façon indépendante des fichiers et du modèle de l'application. Des lecteurs/écrivains travaillent uniquement entre le conteneur et un ou plusieurs fichiers, ils sont donc complètement indépendants du modèle d'application. Cette architecture permet la réutilisabilité des programmes effectuant ces opérations. La figure 6.3 illustre les interactions entre le conteneur et divers types de fichiers.

Les imports/exports entre le conteneur et les fichiers sont gérés au niveau du framework. Si un format de fichier n'est pas géré, le développeur peut alors créer cette fonctionnalité pour son application, et éventuellement ensuite proposer son intégration dans le framework. A terme

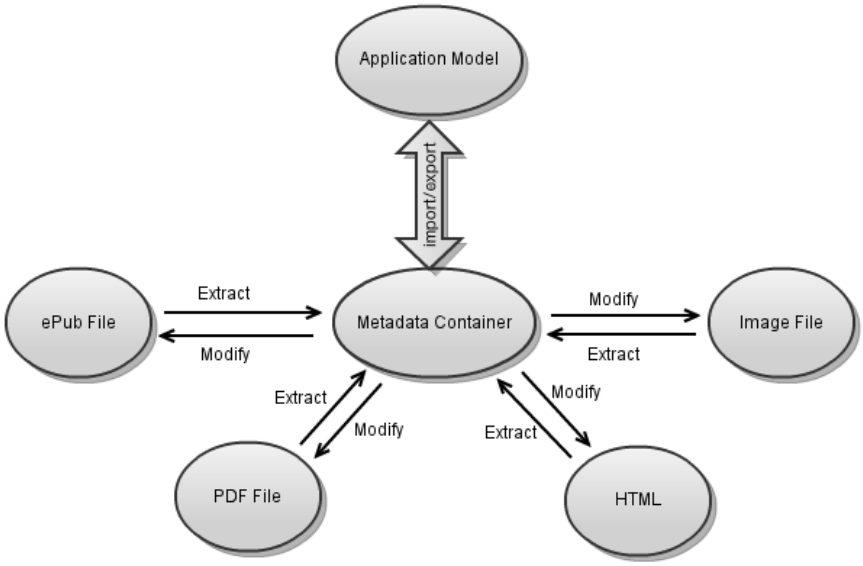

FIGURE 6.3 – Interactions entre le conteneur de métadonnées et divers types de fichiers

cependant, les formats de fichiers les plus courants seront gérés par le cœur de Sydonie. Notons que ces fonctions d'import/export ont vocation à utiliser des bibliothèques tierces pour interagir avec les fichiers. Par exemple, l'import/export dans les fichiers images est réalisé à l'aide de la bibliothèque ExifTool [65], utilisée entre autres par le site d'images Flickr.

Le conteneur décorrèle complètement la gestion de la lecture/écriture d'informations dans un fichier de la gestion des schémas de métadonnées utilisés. Certains schémas de métadonnées supportés par l'import/export sont définis par défaut dans le moteur de Sydonie. C'est le cas du Dublin Core, Exif, XMP. Cependant, lorsque des éléments d'autres schémas de métadonnées sont nécessaires, l'application peut spécifier les éléments qui seront gérés par les procédures d'import/export.

Le conteneur de métadonnées groupe les informations extraites par schéma utilisé, mais les informations à ce niveau sont complètement indépendantes du concept de document de Sydonie. La figure 6.4 donne un exemple de la structure des données pour un document PDF. Celui-ci contient des métadonnées natives au format PDF et des métadonnées incluses avec XMP. L'exemple suppose que les données XMP contiennent des informations du schéma Dublin Core et du schéma LOM (Learning Object Metadata) [HODGINS & DUVAL 02], dont l'objectif est la description de ressources éducatives. L'exemple illustré par la figure 6.4 montre de plus que des informations identiques peuvent être présentes dans plusieurs schémas.

L'étape suivante du processus interne à Sydonie consiste à établir les correspondances entre les métadonnées que le conteneur stocke et le modèle de document de l'application.

65. http://www.sno.phy.queensu.ca/~phil/exiftool/

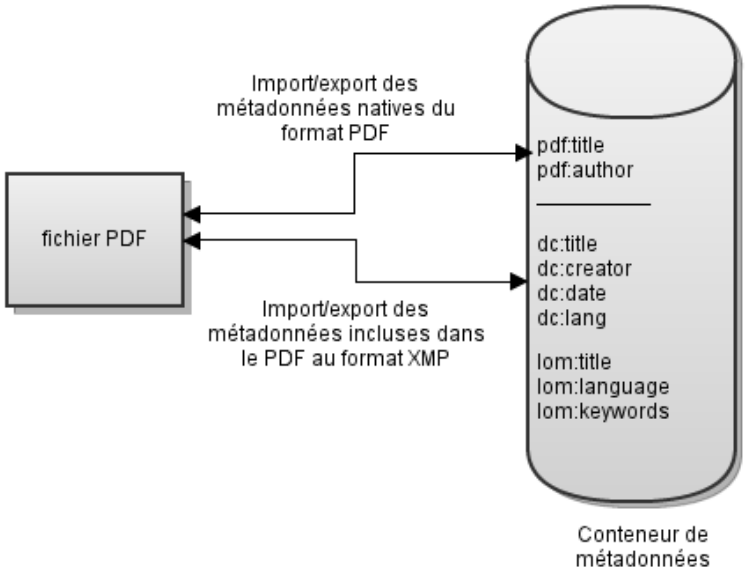

FIGURE 6.4 – Structure de données du conteneur de métadonnées

6.4 *Mappings* métadonnées/modèle de l'application

Dans le conteneur de métadonnées, les informations sont groupées par schémas de métadonnées. Il y a donc possibilité de redondances si plusieurs schémas sont utilisés, comme le montre l'exemple de la figure 6.4. De plus, la vue sur ces informations est *à plat*. La ressource décrite étant une Manifestation, les métadonnées exprimées dans le conteneur correspondent à la fusion des métadonnées prises en remontant les nœuds parents de cette Manifestation.

Le système doit donc effectuer une correspondance entre ces données et le modèle de l'application. Le *mapping* doit être défini par le développeur en fonction des données disponibles et du modèle de l'application. Il doit préciser à quel niveau dans l'arborescence du document (Work, Expression, Manifestation) les informations se situent. La figure 6.5 illustre cette étape.

Le développeur définit les procédures qui établissent les correspondances entre les données. Le processus peut alors appliquer des conversions à certaines valeurs lors du *mapping* (dates, coordonnées GPS par exemple). Dans le cas de redondances lors de l'extraction, comme dans l'exemple de la figure, le développeur définit le champ qui sera prioritaire. Le processus étant par la suite validé par un utilisateur, celui-ci pourra effectuer les corrections nécessaires. En revanche, l'écriture des métadonnées dans le fichier assure une cohérence des informations entre les divers schémas de métadonnées. Dans le cas de types de fichiers « classiques » comme les images par exemple, le framework fournit des procédures par défaut pour la création des documents de type Image. Ces procédures par défaut peuvent être modifiées dans l'application si besoin.

Il est de la responsabilité du développeur de vérifier la cohérence de l'ensemble du pro-

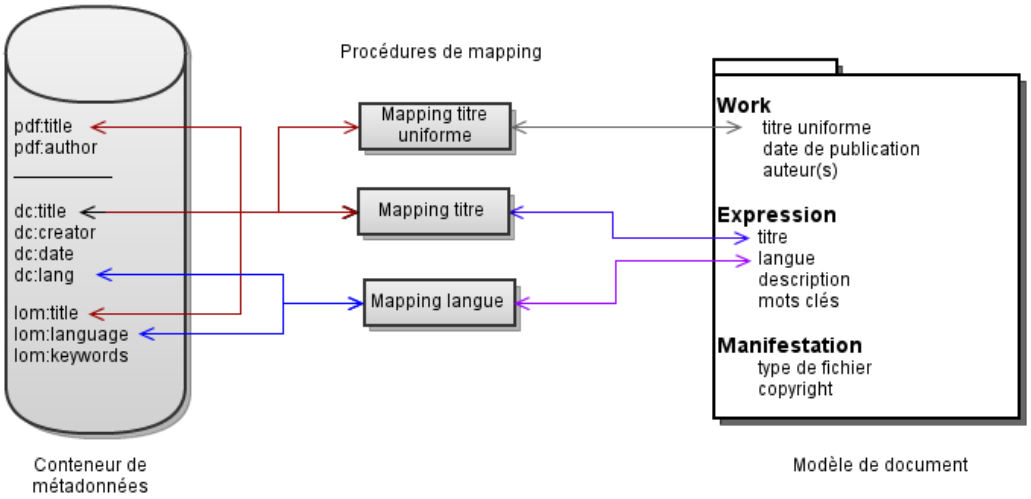

FIGURE 6.5 – Définitions des procédures de mapping

cessus et de créer les procédures de *mapping* en gardant à l'esprit les critères de qualité des données. En particulier, Bruce et Hillman [BRUCE & HILLMANN 04] indiquent que les éléments de métadonnées choisis doivent décrire les objets de façon complète mais réaliste. Le développeur doit donc proposer les éléments qui sont essentiels pour le public visé, mais sans inclure une masse d'informations dont la plupart ne sera jamais renseignée ou utilisée.

6.5 Processus complet, cohérence des données et IHM

Les interfaces de validation et de saisie des informations sont créées pour chaque application. Là encore, Sydonie peut fournir des interfaces par défaut comme c'est le cas pour les images par exemple. Les formulaires peuvent donc être pré-remplis avec les métadonnées extraites du fichier lui-même. L'utilisation du modèle de document de Sydonie assure la cohérence des données lors de l'ajout d'une nouvelle Expression ou Manifestation et évite la duplication. La figure 6.6 illustre, dans le cadre de l'application `lisez-moi.lu`, le processus de création d'un eBook lors de l'upload d'un fichier ePub.

Les étapes 1 et 7 montrent la couche d'abstraction implémentée avec le conteneur de métadonnées détaillé à la section 6.3. Ces étapes sont intégrées au moteur du framework Sydonie. Leur objectif est d'assurer, du point de vue informatique, l'accessibilité et la cohérence des données. Ces étapes sont réalisées par les lecteurs/écrivains qui gèrent les import/export dans divers types de fichiers. Ces programmes sont typiquement créés par l'équipe de Sydonie ou par des développeurs indépendants qui les mettent à disposition de la communauté. Ils sont destinés à être intégrés dans le moteur de Sydonie et à faire appel à des bibliothèques externes.

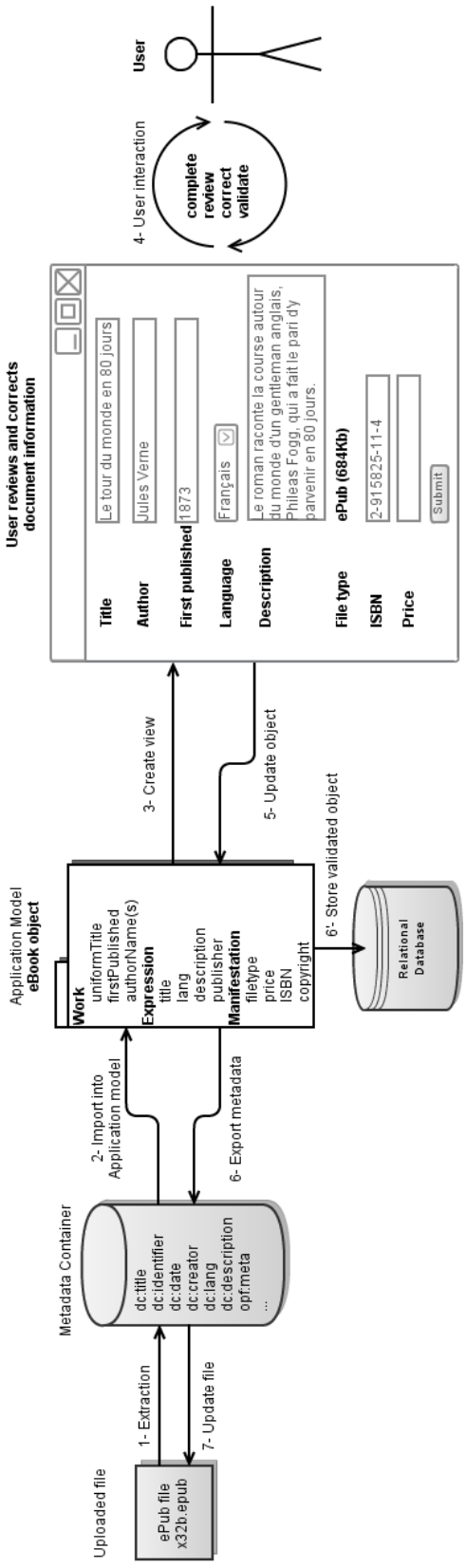

FIGURE 6.6 – Processus de création d'un eBook à partir d'un fichier ePub

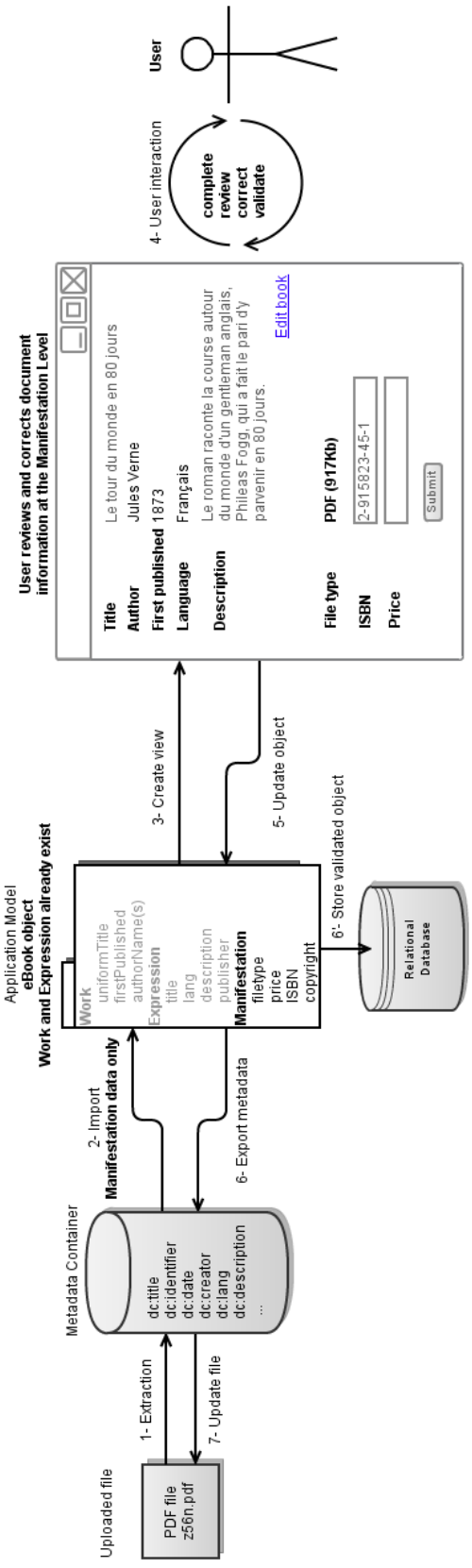

FIGURE 6.7 – Procédure d'ajout d'un PDF pour un eBook existant

Les étapes 2 et 6 montrent les procédures de *mapping* entre les données du conteneur et le modèle de document défini pour l'application. Cette étape est réalisée par le développeur web. il définit les correspondances à effectuer entre les données du conteneur et le modèle de son application.

Les étapes 3, 4 et 5 illustrent les interactions entre le modèle de l'application, les vues créées et l'utilisateur. Ces interactions sont réalisées par le développeur web avec le graphiste ou l'ergonome. La définition des processus de validation doit prendre en compte les besoins des utilisateurs de l'application. Les interactions avec l'utilisateur (étape 4) sont importantes pour la complétude et la précision des données et des métadonnées. Le producteur d'information (en l'occurrence un éditeur dans le cadre de `lisez-moi.lu`) est responsable de la validation des informations qui ont été extraites.

Lors de l'ajout d'un nouveau format pour un eBook existant, illustré figure 6.7, le modèle de document de Sydonie montre ses avantages, permettant ici d'utiliser les informations déjà présentes dans le système pour les niveaux Work et Expression. Seules les informations concernant le fichier ajouté, c'est-à-dire au niveau de la Manifestation, doivent alors être validées et complétées par l'éditeur.

Le travail du développeur est finalement facilité. Grâce à la couche d'abstraction fournie par le conteneur de métadonnées et les routines d'import/export existantes, il peut désormais se concentrer sur les procédures de correspondance entre les données du modèle de document à gérer et les métadonnées. L'utilisation du modèle de document de Sydonie permet d'assurer la cohérence des informations contenues dans divers fichiers. De plus, la gestion des interactions permet la construction de formulaires adaptés et la gestion des informations aux différents niveaux Work, Expression et Manifestation. La notion d'arbre FRBR et de schéma de métadonnées deviennent alors transparentes pour l'utilisateur final, qui en général n'est pas spécialiste de la gestion d'information. Les données saisies par l'utilisateur sont alors transformées en métadonnées grâce aux processus fournis par Sydonie.

6.6 Discussion

Les procédures et méthodes présentées dans ce chapitre fonctionnent bien pour les fichiers incluant des métadonnées, tels que les images, les fichiers PDF, ePub ou HTML. L'ajout d'autres types de fichiers gérés par Sydonie se fera au fil du temps et des contributions de divers projets.

La définition des correspondances implique pour l'instant que le développeur connaisse les éléments des schémas de métadonnées utilisés. Il sera donc nécessaire d'ajouter à des interfaces facilement utilisables pour la définition du mapping entre les données du modèle et les métadonnées.

Nous réfléchissons à l'utilisation d'une entité Manifestation spécifique au format XML/RDF,

qui exprimerait les métadonnées et pourrait répondre à des requêtes émanant de moteurs ou d'outils spécifiques.

D'autres travaux commencent pour expérimenter l'utilisation de méthodes de marquage de texte avec l'utilisation de la TEI. Ces travaux, menés par Pierre-Yves Buard [Dornier & Buard 08] dans le cadre d'une thèse, permettront d'améliorer les services proposés par le framework Sydonie.

Les développeurs qui ont uilisé ce système ont apprécié la couche d'abstraction apportée par le framework. Leur travail a alors pu se concentrer sur la gestion des interactions des utilisateurs avec l'application.