Rapport de recherche sur les forges de publication

Roch Delannay, Antoine Fauchié

RAPPORT DE RECHERCHE SUR LES FORGES DE PUBLICATION

Version v1.1 - 2024-12-03

Sous la direction d'Emmanuel Château-Dutier, Michael Sinatra et Marcello Vitali-Rosati

Coordonné et écrit par Roch Delannay et Antoine Fauchié

TABLE DES MATIÈRES

AVANT-PROPOS

Ce rapport a été commandité à la Chaire de recherche du Canada sur les écritures numériques (CRCEN) par le groupe de recherche et réflexion CIÉCO. Ce rapport aborde les forges d'édition et de publication, ou *chaînes d'édition multi-formats*, et participe à une réflexion plus générale sur les nouvelles formes de valorisation des collections muséales et d'interactions avec différents publics. Il est le fruit du plusieurs mois de travail, d'échanges, de recherche et d'une étroite collaboration entre les auteurs, Antoine Fauchié et Roch Delannay, et les relecteurs et les relectrices qui ont donné de leur temps pour permettre une finalisation de cette première version du texte.

Nous tenons à remercier tout spécialement Emmanuel Château-Dutier, professeur au département d'histoire de l'art, de cinéma et des médias audiovisuels à l'Université de Montréal et qui est à l'origine de ce rapport ; Michael Sinatra, professeur au département de littératures et de langues du monde et directeur du Centre de recherche interuniversitaire sur les humanités numériques ; Marcello Vitali-Rosati, professeur au département des littératures de langue française et titulaire de la CRCEN ; Mathilde Verstraete, doctorante en littérature option humanités numériques à l'Université de Montréal ; et Louis-Olivier Brassard, doctorant en littérature option humanités numériques à l'Université de Montréal.

Le groupe de recherche et réflexion CIÉCO est piloté par une équipe de chercheurs provenant à la fois de l'Université de Montréal, mais aussi de l'Université du Québec à Montréal, du Musée d'art de Joliette, du Musée des beaux-arts de Montréal et du Musée national des beaux-arts du Québec. Il fédère ainsi les deux institutions de la discipline de l'histoire de l'art : l'université et le musée. Ce partenariat s'articule autour de la thématique de la mise en valeur des collections muséales, qui ont longtemps été concurrencées par les expositions. Les équipes qui travaillent sur ce projet ont pour ambition de repenser le format de la collection muséale autour des enjeux liés à la numérisation et à la diffusion sur le Web.

Roch Delannay (carnet.en-cours-de.construction) est doctorant à l'Université de Montréal (Canada), et Antoine Fauchié (www.quaternum.net) est post-doctorant à l'Université de Rouen Normandie (France) après un doctorat à l'Université de Montréal (Canada).
Leurs recherches gravitent autour des techniques d'écriture, d'édition et de publication dans des environnements numériques. Elles portent respectivement sur les différents processus d'écriture scientifique et sur les fabriques d'édition numérique. En parallèle de ces travaux, ils sont également impliqués dans différents projets de recherche de la CRCEN tels que l'éditeur de texte Stylo, le Partenariat Revue3.0, ou encore la chaîne d'édition le Pressoir.

Ce rapport a un double enjeu : tout d'abord répondre aux objectifs scientifiques du groupe CIÉCO en matière de valorisation et de médiation, mais également produire une preuve de concept qui s'inscrit dans le champ de l'édition numérique. Le texte de ce rapport repose sur le principe de *single source publishing* et fait l'objet d'une édition multi-formats. La forge que nous avons créé pour cette occasion repose sur les formats suivants : Markdown pour le texte, les métadonnées sont sérialisées en YAML, et les références bibliographiques sont structurées en BibTeX. Toutes ces données textuelles sont ensuite converties et transformées par le générateur de site statique Eleventy pour produire la version web du rapport et une version HTML paginée pour l'impression. La version PDF est générée avec la bibliothèque JavaScript Paged.js. Le rapport n'est disponible en accès libre qu'au format PDF, la version web sera mise à disposition ultérieurement.

INTRODUCTION

UN RAPPORT SUR LES CHAÎNES DE PUBLICATION EN CONTEXTE ACADÉMIQUE

Dans le cadre de la préparation et de la réalisation d'une encyclopédie en sciences humaines, et plus spécifiquement en histoire de l'art, le groupe de recherche Collections et impératif événementiel/The Convulsive Collections (CIÉCO) réalise une étude sur les chaînes de publication en contexte académique. Il s'agit de repérer et de décrire des solutions pour l'édition numérique multimodale sémantique : ces *chaînes de publication* prennent en charge la structuration des contenus, la conversion de fichiers sources balisés dans des *formats* divers, et l'organisation de ces contenus structurés en vue de les éditorialiser. L'objectif de ces outillages techniques est de transformer des données richement structurées pour obtenir des objets numériques comme des sites web, des jeux de données (API, *JSON*, CSV, etc.), des contenus balisés (*HTML*, *XML*, etc.), des applications, mais aussi des formats destinés à l'impression. L'enjeu de ces chaînes de publication est autant la prise en compte de données riches, la conversion vers des formats consultables ou réutilisables, que l'organisation des contenus et des données sur des plateformes accessibles.

Les solutions existantes sont décrites et critiquées selon des critères clairement exposés, et notamment l'interopérabilité, l'ouverture du code, la modularité des briques techniques ou l'usage de standards. Les cadres d'usage sont présentés et détaillés afin de comprendre les enjeux théoriques dans des implémentations pratiques. Les limites inhérentes de ces technologies sont confrontées aux exigences de la publication scientifique en général et du projet de CIÉCO en particulier. L'objectif est de décrire également des briques techniques qui répondent à certaines des ambitions du projet éditorial du CIÉCO, et principalement la conservation d'une richesse sémantique, afin de proposer des pistes pour l'établissement d'une chaîne de publication reproductible.

Ce rapport comprend un état de l'art — par définition synthétique et non exhaustif —, des grilles d'analyse basées sur des critères présentés et argumentés, des fiches descriptives de solutions technologiques et d'usages, et des mises en perspective théoriques, pratiques et critiques.

CONTEXTUALISATION

Le premier constat est le suivant : au début des années 2020 les environnements de publication pour les données textuelles et médiatiques sont riches et de plus en plus

interopérables. Les données peuvent enfin être réutilisables puisqu'elles sont bien souvent structurées, liées à des schémas et documentées. Il y a une forme de renouveau documentaire, en effet les documents peuvent désormais être édités, convertis ou transformés en utilisant des standards et en respectant les principes de l'interopérabilité. C'est dans ce cadre que les solutions pour administrer les données et tenter de les valoriser se multiplient. Le panorama des outils de production et de valorisation est pourtant relativement restreint, en effet le second constat est que la plupart des solutions ne sont que peu ou pas adaptables ou au prix d'efforts et de coûts très importants. En d'autres termes, chaque solution technique répond à des objectifs précis, et il est souvent difficile de modifier le comportement de ces solutions pour les adapter à des besoins différents ou génériques. Par ailleurs, si les données en entrée doivent être sémantiquement riches et structurellement interopérables, encore faut-il que l'outil qui les traite conserve ces deux propriétés — à l'intérieur comme format pivot et en sortie comme artefact. Les réponses techniques consistant à intégrer un nombre important de fonctionnalités dans une même application ne sont pas toujours adéquates. En effet les cas d'usage sont soit trop larges et alors il est difficile de disposer d'options suffisamment précises, soit trop restreints et alors ces solutions sont limitées à des situations trop restrictives et difficilement duplicables dans d'autres contextes.

Les méthodes et les processus existants et utilisés pour la publication scientifique sont nombreux, depuis les débuts de l'informatique et plus encore depuis le début des années 2000 et l'avènement du numérique. Différentes solutions techniques sont mises en place pour répondre aux exigences de l'édition scientifique tout en prenant en compte les différentes contraintes liées à la nature des données en entrée de la chaîne de publication. Mais qu'est-ce qu'une *chaîne de publication* ? Il s'agit de l'ensemble des méthodes et des processus, y compris leur implémentation technique, consistant à produire des publications. Cette définition générique prend en compte, notamment, les phases d'inscription, de soumission, de relecture, de correction, de conversion, de transformation, de mise en forme ou de production des documents finaux. Nous explorons ici un certain nombre de chaînes de publication, plus ou moins complètes : certaines sont des technologies de l'édition numérique (Blanc & Haute, 2018), ou des technologies numériques de l'édition, et permettent de composer des chaînes de publication.

Dans le contexte d'un certain *renouveau documentaire* mentionné plus haut, un mouvement ou une tendance particulière doit être mentionnée : la *JAMStack* (Markovic & Scekic, 2021) — pour JavaScript, API and Markup Stack — est une réponse technique à une complexité grandissante des systèmes de gestion des contenus. Au cœur de cette pratique issue du développement informatique (côté Web) des outils d'un nouveau genre ont fait

leur apparition au début des années 2010 : les générateurs de site statique. Ces initiatives privilégient la mise à plat des données, préférant des contenus balisés à des bases de données relationnelles complexes à maintenir et gourmandes en ressources. C'est donc après le *boum* des CMS que les générateurs de site statique s'imposent. Mais quel intérêt pour les sciences humaines et sociales et plus encore pour l'édition scientifique ? Travailler avec des architectures découplées — donc en partie moins monolithiques — ouvrent des possibilités intéressantes, comme la pérennisation des données, mais aussi des environnements de travail et de développement plus adaptatifs.

Nous utilisons ici le terme de *forge* pour définir des environnements de conception, de fabrication et de production pour des publications scientifiques. Ce terme nous permet de placer sur un même plan des solutions complètes de publication, des briques techniques spécifiques comme des convertisseurs de formats, ou encore des processeurs de documents. Le terme *forge* intègre deux dimensions qui répondent au spectre d'étude de ce rapport : la notion de *fabrication*, qui correspond ici au fait de *fabriquer* des objets éditoriaux ; la notion d'*atelier*, qui comprend le fait de travailler à plusieurs dans un espace défini où se fabriquent des objets. Le terme *forge* implique également la notion d'artisanat, avec cette idée d'une maîtrise technique et précise afin de réaliser un travail de qualité.

CADRE THÉORIQUE ET PRATIQUE

Le présent rapport a pour vocation de traiter partiellement la question de l'éditorialisation (Vitali-Rosati, 2016) pérenne de collections de documents. La portée de cette question est très vaste, et proche de certaines problématiques liées à l'archivage, nous ne la traitons toutefois pas selon cette perspective. L'angle adopté est celui des enjeux des logiciels et des données. En effet, l'objet de ce rapport ne concerne pas les couches matérielles — *hardware* — et supports, il est centré sur des chaînes logicielles de publication de textes et de données. Ces chaînes ont comme source des collections de documents structurés, et pour sortie ces collections transformées selon les différentes modalités de diffusion.

L'éditorialisation est un concept central de ce rapport que nous pouvons définir comme :

> […] l'ensemble des dynamiques qui constituent l'espace numérique et qui permettent à partir de cette constitution l'émergence du sens. Ces dynamiques sont le résultat de forces et d'actions différentes qui déterminent après-

> coup l’apparition et l’identification d’objets particuliers (personnes, communautés, algorithmes, plateformes…). (Vitali-Rosati, 2021).

Cet “espace numérique” pose un certain nombre de contraintes techniques, par exemple lorsqu’il s’agit de pérenniser des contenus dans une infosphère qui évolue rapidement. L’obsolescence qui en découle menace la lisibilité des documents numériques, qu’ils soient nativement numériques ou numérisés, incorporant une sémantique riche ou pauvre. Afin de répondre aux problématiques posées dans le cadre du projet de recherche CIÉCO, nous concentrons ce rapport sur les forges de publication liées aux technologies du Web. L’un des aspects les plus importants de ce rapport repose sur la nécessité de rendre visible des collections de documents. Ainsi, les besoins de visibilité, de pérennité, d’ouverture la plus large possible (standards et libre/*open source*) précisent le champ d’action du rapport au Web et à ses technologies. Le terme Web utilisé ici englobe les différentes strates qui la composent telles que les différents protocoles et standards de communication, les langages de balisage, les procédés de standardisation et d’ouverture afin d’assurer la rétrocompatibilité et l’interopérabilité des contenus. Cette partie est développée plus en détail lors de la présentation des différentes forges.

Nous employons régulièrement la notion de *document* tout au long de ce rapport. Le document est à comprendre selon son sens le plus large et ne représente pas seulement un fichier texte ou une *feuille volante* déposée sur le coin d’un bureau. Un document (numérique) (Ihadjadene, Zacklad, & Zreik, 2010) est un espace d’organisation de l’information dépendamment d’un schème précis : il peut s’agir d’un texte, d’un script, d’une image, etc. Nous pouvons aussi citer les trois dimensions du document telles que formulées par Roger Pédauque : forme, signe et médium (Pédauque & Melot, 2006).

Critères de sélection et grille d’analyse

Afin de choisir puis d’étudier plusieurs solutions technologiques, il faut établir des critères de sélection ainsi qu’une grille d’analyse. Ces éléments sont détaillés dans les parties suivantes du rapport. Ces critères et cette grille présentent un intérêt majeur, puisqu’ils déterminent notre champ d’étude et la façon de la mener.

Les critères de sélection répondent aux besoins du projet de CIÉCO — à savoir une chaîne de publication multimodale basée sur des logiciels libres pour produire une encyclopédie ou une collection d’éléments sous la forme d’une publication aux formats numériques et imprimés — mais ces critères peuvent aussi être considérés d’un point de vue plus global et correspondre à d’autres projets de publication. Il s’agit tout

d’abord de prendre en considération la nature des solutions. Nous souhaitons analyser des chaînes de publication complètes, prenant en compte autant la production que l’organisation des contenus. Ainsi les chaînes de publication, les générateurs de sites web, les convertisseurs mais aussi les processeurs de document peuvent entrer dans notre périmètre. Le critère suivant est le caractère libre ou *open source* des outils disponibles : la licence associée au logiciel ou à la bibliothèque de code doit permettre des utilisations ou des modifications aussi libres que possibles. L’usage des standards est un autre critère déterminant, mais néanmoins sujet à interprétation. Certaines initiatives utilisent des standards *établis* — au sens d’une série de pratiques documentées et issues de décisions collégiales —, mais il s’agit parfois de standards *de fait* — c’est-à-dire en tant que résultat d’une adhésion large et partagée mais sans structure/organisation de documentation et de validation collectives. Ensuite, la dimension modulaire est un point important pour permettre un assemblage et une reconfiguration des solutions techniques entre elles. Les solutions présentées dans ce rapport ne doivent pas forcément répondre à tous ces critères, mais au moins à plusieurs d’entre eux.

La grille d’analyse est mise en place d’abord pour étudier les *forges*, elle permet de structurer les différents points à aborder et à critiquer pour une analyse pertinente :

- descriptions courte et longue permettant de présenter les objectifs et l’historique de la forge ou de la technologie ;
- formats des fichiers en entrée et en sortie de ces outils ;
- usages et exemples permettant de prendre la mesure des cas d’utilisation ;
- technologies utilisées pour faire *fonctionner* cet outil, aussi appelées *dépendances* ;
- les possibles interfaçages avec des interfaces graphiques pour éditer les contenus, organiser l’ensemble des informations ou modifier les paramètres ;
- contraintes : l’ensemble des règles d’utilisation comme les formats de balisage ou leurs spécificités, des limitations comme des environnements informatiques ou encore des freins à leur utilisation.

PLAN DÉTAILLÉ

Pour répondre aux enjeux de ce rapport nous articulons les différentes parties de la manière suivante : la première partie de ce rapport concerne un état de l’art des forges. Il existe une myriade de solutions possibles pour répondre aux enjeux qui nous concernent et nous n’avons pas vocation à toutes les cataloguer. Toutefois, nous pens-

ons que certaines de ces forges présentent une ou plusieurs spécificités méritant d'être énoncées, ne serait-ce que pour positionner cette étude dans cette étendue de technologies. Une seconde partie est centrée sur l'identification des différentes contraintes d'usages, contraintes techniques ou sociales et plus généralement des *workflows* induits par les forges. Comme nous l'avons déjà mentionné, les enjeux de pérennité et de préservation de la richesse sémantique des documents impliquent des contraintes, comme par exemple la contrainte du *format* des documents entrant et sortant des forges, qui seront révélées par l'état de l'art. Une fois cette phase achevée, nous présentons une sélection des forges retenues. Afin d'illustrer au mieux les choix des forges vis-à-vis des critères d'analyse mentionnés, nous avons créé des fiches permettant de mettre en exergue les éléments qui répondent à ces critères en limitant les explications des différents fonctionnements des outils. Avant de conclure, une quatrième partie est consacrée à une mise en perspective des contraintes mentionnées dans les fiches des forges et les contraintes relatives aux problématiques de mise en valeur des collections de documents. Ensuite, et dans la même optique que la partie précédente, nous faisons une synthèse des forges présentées en regard avec l'état de l'art ; puis nous abordons différentes solutions envisagées pour contourner les contraintes ou problèmes rencontrés. Enfin, la dernière partie est dédiée à un exemple d'usage de cette étude dans un contexte de publication scientifique en sciences humaines.

RÉFÉRENCES

Blanc, J., & Haute, L. (2018). Technologies de l'édition numérique. *Sciences du design*, *8*(2), 11-17.

Ihadjadene, M., Zacklad, M., & Zreik, K. (2010). *Document numérique entre permanence et mutations*. Éditions Europia, Paris.

Markovic, D., & Scekic, M. (2021). *Understanding Jamstack and Its Perception in Web Development*.

Pédauque, R. T., & Melot, M. (2006). *Le document à la lumière du numérique* (J.-M. Salaün, Éd.). Caen, France: C&F éditions.

Vitali-Rosati, M. (2016). Qu'est-Ce Que l'éditorialisation ? *Sens Public*.

Vitali-Rosati, M. (2021). Pour une pensée préhumaine. *Sens public*, (SP1596).

ÉTAT DE L'ART

Dans cette partie nous proposons de faire un état de l'art des forges de publication. Comme nous l'avons mentionné en introduction, le terme *forge* est employé pour sa propriété polysémique. Il désigne dans un premier temps un atelier numérique, un espace collaboratif duquel émergent des réflexions sur les procédés, les techniques et les développements possibles en réponses à des besoins. Dans un deuxième temps, la forge est un espace de fabrication où, à travers différents processus de transformation, des matériaux bruts sont transformés en artefacts. Nous utilisons le terme forge parce qu'il recouvre ici les notions d'espace collaboratif, de transformation et de conversion. Nous désignons par *forge* les différentes étapes, procédés et règles qui permettent l'inscription de contenus, la conversion, la transformation et l'organisation des fichiers d'un format à un autre format.

FONCTIONNEMENT D'UNE CHAÎNE GÉNÉRIQUE

Lors de cette première partie de l'état de l'art nous présentons une forge générique afin de détailler les différents éléments qui la composent. Par *forge générique*, nous considérons qu'il est possible de conceptualiser le fonctionnement d'une chaîne de publication. Nous devons préciser ici que ce rapport s'intéresse plus particulièrement à l'approche dite du *Single Source Publishing*, sans pour autant laisser de côté d'autres manières de procéder.

Single Source Publishing : une source unique pour produire différents artefacts

Une précision s'impose : l'approche dite du *Single Source Publishing* — traduite par publication à partir d'une source unique — est une approche privilégiée dans certaines solutions ou forges présentées ou analysées, mais pas dans toutes. Ainsi, nous ne présentons pas uniquement des démarches qui suivent ce principe, dressant un panorama aussi large que possible.

La publication ou l'édition doivent relever un certain nombre de défis, comme la production de plusieurs artefacts à partir d'une même source. Pourquoi s'intéresser à ce qui semble être une démarche parmi d'autres ? L'expression *Single Source Publishing* désigne le fait de générer plusieurs formats à partir d'une seule et même source (Fauchié & Audin, 2023; Hyde, 2021). Un seul et unique document permet de produire des formats divers, sans avoir besoin de basculer d'une version de travail à une autre. Que ce

soit un format PDF pour l'impression, un export XML pour un diffuseur numérique ou une version numérique au format HTML, l'objectif est de travailler avec une même source.

Cette source n'est pas forcément un seul fichier, mais peut comprendre des fichiers texte, des *médias* comme des images, des fichiers sonores ou des vidéos, mais aussi des données structurées de différentes façons. Il s'agit d'un enjeu éditorial qui soulève des questions autant théoriques que techniques, telles que la légitimation des contenus, l'évolution des pratiques d'édition ou la création d'outils adéquats, que nous ne détaillerons pas ici.

Le *Single Source Publishing* est une problématique autant pour les éditeurs que pour les distributeurs/diffuseurs, car il permet de produire autant de formats que nécessaire tout en conservant une même origine. Les apports sont nombreux :

- création d'un espace commun pour tous les acteurs de la chaîne d'édition ;
- clarification des interventions sur les contenus ;
- croisement des besoins entre les différents formats de sortie ;
- mutualisation des efforts pour les différents exports à produire (PDF, HTML/ Web, XML, EPUB, etc.) ;
- simplification de l'archivage en ne disposant que d'une seule et même source.

La mise en place de chaînes de publication implémentant le *Single Source Publishing* est complexe, plusieurs initiatives tentent de réaliser ce défi depuis les débuts de l'informatique. Les implémentations sont nombreuses, et respectent de façon plus ou moins approfondie les contraintes imposées par cette approche : LaTeX, Métopes, Stylo, Quire, Manifold, etc.

Deux niveaux : organisation et implémentation technique

À une échelle *méta*, la constitution d'une forge combine deux niveaux indissociables : un niveau organisationnel ou humain, et un autre technique. Ces deux niveaux sont imbriqués et ne peuvent pas être séparés de façon hermétique. La partie sur laquelle nous allons nous concentrer sera principalement la seconde couche : l'implémentation technique. C'est-à-dire que nous n'aborderons pas en détail les gestes d'écriture, de tri, de vérification et de correction que nécessite tout protocole d'édition. Toutefois, nous préciserons par des annotations les actions humaines lorsque ce sera nécessaire.

Schéma

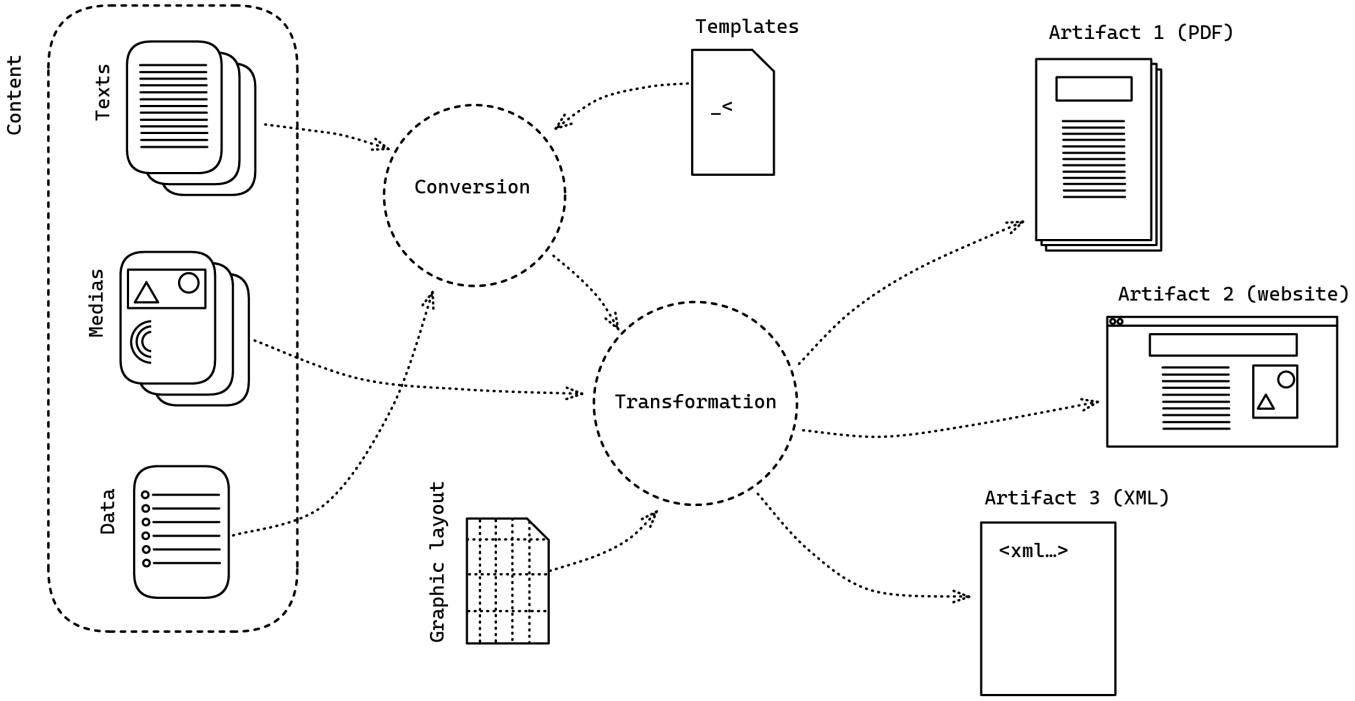

Figure 1. Schéma complet d’une chaîne de publication générique

Dans ce schéma nous observons quatre opérations principales :

1. l’écriture ou l’inscription des documents, comme des textes, des médias ou des données ;
2. la conversion de ces documents, notamment d’un format de balisage à un autre format de balisage, mais aussi en réorganisant les fichiers ;
3. la transformation d’où résulte la production des artefacts ou des formats de sortie, comme les formats PDF, HTML ou XML ;
4. la publication des artefacts.

Écriture et inscription

La question de l’écriture ou de l’inscription ne concerne pas uniquement du texte mais toutes les formes d’inscription possibles sur un support numérique. Il peut s’agir de l’écriture d’un texte, d’une photo, d’une collection de documents numérisés, d’un document sonore, etc. Ces inscriptions seront nos **sources primaires**, pour lesquelles nous souhaitons conserver toute la richesse sémantique. Cette étape de préservation et de conservation est importante car le risque majeur encouru par ces documents est la dilution sémantique après de multiples étapes de transformations et de conversions des

documents dans d’autres formats adaptés aux artefacts souhaités. Chaque format impose une architecture spécifique pour les données contenues dans les documents et, du fait de ces différences entre les structures de données, il y a un fort risque de perte d’information lors du passage de l’un à l’autre pour correspondre aux exigences des artefacts à produire.

Les enjeux de cette étape d’écriture consistent en un enrichissement sémantique maximal des sources primaires puis de la préservation des informations en contournant les pertes engendrées par le passage d’un format à un autre format. Par exemple, dans le cas d’une transformation d’un document dont le contenu est encodé dans un format à plusieurs niveaux d’informations vers un format qui n’en propose qu’un seul (en colonne par exemple), il devient nécessaire de trouver des procédés de structuration horizontaux des informations (sur un seul niveau) afin de ne pas perdre la richesse de la structure du premier format. Si cette réflexion n’avait pas lieu, toutes les informations se retrouveraient à un même niveau hiérarchique et leur lecture et interprétation en seraient erronées.

Conversion

Convertir et transformer un document sont deux opérations distinctes. La conversion et la transformation sont des processus qui requièrent l’application d’un modèle de structuration des écritures numériques aux sources primaires. Ces opérations ne sont pas similaires, notamment du fait de la propriété de réversibilité du processus de conversion et non de celui de transformation.

La conversion est l’application de règles permettant de passer d’une façon de structurer l’information à une autre. Concrètement, cela se traduit par un changement de balisage. Une chaîne de publication comporte plusieurs conversions qui forment des processus complexes.

La conversion d’un document peut se faire dans les deux sens, c’est-à-dire depuis la source vers le format final ou inversement, mais seulement si les modèles de conversion sont disponibles. En d’autres termes, s’il y a une perte d’information lors de la conversion, il ne sera évidemment pas possible de faire le processus inverse. Ainsi la conversion est un processus réversible selon certaines conditions.

Transformation/production

Au contraire, la transformation est un processus généralement unilatéral sans aucun moyen de revenir à l'état initial de la source primaire, comme par exemple avec la génération du format PDF. C'est pour cela que nous nommons également cette étape *production* : il s'agit de *produire* un format qui ne sera ni source ni pivot, mais uniquement un format mis à disposition. Autrement dit, cette production implique de ne pas réintroduire un nouveau fichier dans la chaîne mais uniquement de l'exporter sans nouvelle manipulation. Dès lors, il devient primordial de définir ces différents processus en fonction des types d'artefacts à produire. Un objet tel qu'un livre imprimé sera forcément pensé différemment qu'un site Web.

Qu'en est-il si les productions sont multiples ? Deux réponses génériques peuvent être apportées à cette question. Il est possible de séparer les processus où chaque artefact correspond à un flux de travail précis. Ainsi une conversion peut être créée uniquement pour une transformation spécifique. Voici un exemple concret : la *conversion* du format Markdown vers le format TEX permet ensuite une *transformation* du format TEX au format PDF. La conversion et la transformation sont ici liées.

L'autre réponse est de ne créer qu'un seul flux de travail permettant de générer différents types d'artefacts. Ce besoin a déjà été exprimé : des forges existent sur ce modèle de pensée et reposent sur le principe du *Single Source Publishing* comme nous l'avons déjà mentionné précédemment. À partir du principe de *Single Source Publishing* il est possible d'agencer différents outils pour convertir et transformer une source primaire en différents artefacts. On agrège ensemble différents fichiers pour que la source primaire soit enrichie autant que possible, puis on lui applique les différents modèles qui correspondent aux formats des artefacts à produire.

Format(s) et contraintes

Cependant, le format n'est pas la seule contrainte à anticiper lorsqu'il s'agit d'élaborer une forge. Chaque type d'artefacts impose son lot de contraintes, telle une manière de représenter le monde qui lui est propre. Afin que l'utilisateur accède au contenu d'une publication, il aura la possibilité d'interagir avec le support de cette publication selon des gestes prédéfinis. Les données peuvent donc être gérées selon deux modalités : l'inscription dans les différents formats et la matérialité du support. Ces deux dimensions gagnent à être pensées en parallèle l'une de l'autre car elles sont complémentaires et comblent leurs lacunes mutuellement. Sur un support numérique, un geste comme le clic peut renvoyer à un autre espace dans la page (hyperlien)

et connecter deux informations entre elles pour offrir un lien sémantique entre les deux entités, et compenser ainsi la mise à plat de la source primaire sur le support numérique.

Cette introduction aux forges de publication n'est qu'une vue d'ensemble pour présenter les différentes composantes que l'on y trouve et leur articulation entre elles.

LES ÉTAPES EN DÉTAIL : ÉCRITURE/INSCRIPTION, CONVERSION, TRANSFORMATION/PRODUCTION

Écriture

L'inscription est une partie de l'écriture. Elle consiste en l'action d'écriture de l'information sur le support matériel - *hardware* - que réalise la machine. C'est un paramètre qu'il ne faut pas oublier, l'écriture n'est à aucun moment située dans le logiciel mais bien dans le matériel physique qui stocke les informations (Kittler, 2015). Toutes les opérations qui interviennent par l'entremise des logiciels et des forges ne sont que des suites d'ordres et de commandes envoyées à la machine pour qu'elle procède à cette inscription. Ainsi, toute action réalisée dans le champ du numérique s'apparenterait à une forme d'écriture. Écrire ne relève plus uniquement du geste de tracer des lignes mais devient une application plus globale de l'éditorialisation (Vitali-Rosati, 2016), dans laquelle interviennent des algorithmes, des logiciels, des interfaces, des robots, etc. Chaque événement — *capta* — est inscrit dans une couche différente, dans un paratexte numérique (Genette, 1982; Vitali-Rosati, 2020) qui prend la forme d'un palimpseste (Kembellec, 2020; Mellet, 2020) dont les délimitations permettent l'apparition du document numérique. Le processus d'écriture est donc constant, un pic d'activité est remarquable en début de la chaîne d'une forge, lorsqu'il s'agit de générer les documents à publier : écriture du texte, des images, des sons.

Cette première étape de notre forge générique concerne toute la production des sources primaires au moyen d'outils d'écriture (éditeur et traitement de texte), de traitement d'images, de versionnement, de stockage, de commentaire, etc. Toutefois le processus d'écriture ne s'arrête pas une fois les sources primaires établies. Le paratexte continuera d'accompagner chacune des étapes de la forge jusqu'à la publication finale. En effet, le processus laisse des traces visibles entre les couches telles que des commandes à exécuter (manuellement ou automatiquement) et des mémoires des actions passées (*logs*) qui peuvent être utiles dans la compréhension du processus global et des potent-

iels dysfonctionnements de la chaîne. Cette écriture continue du document participe à sa sémantisation : la lire est une indication de la vision du monde qu'elle porte.

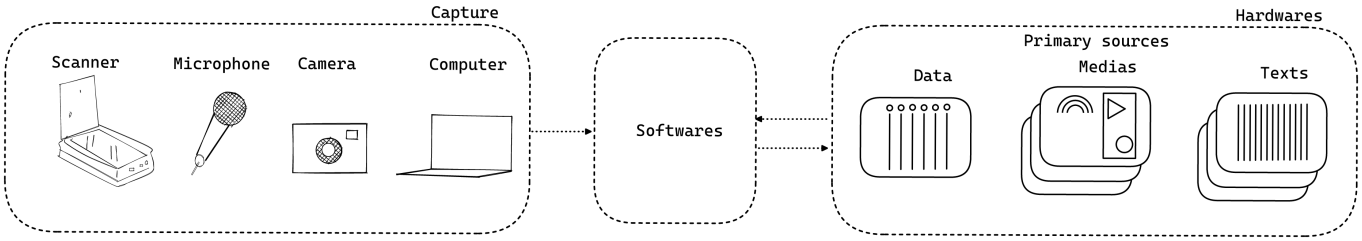

Figure 2. Schéma de la phase d'écriture

Ce schéma nous permet de mettre en image le principe d'écriture que nous venons de décrire. L'écriture débute par une captation d'événements et d'informations, qu'il s'agisse des touches d'un clavier pour du texte, d'un appareil photo pour des images ou des vidéos, d'un microphone pour du son ou d'un scanner pour générer des images de documents qui ne sont nativement pas numériques ; cette liste n'est pas exhaustive. La première flèche, dirigée des capteurs vers les logiciels, désigne le passage des informations par les logiciels afin que celles-ci soient encodées selon un format spécifique au type de document qui leur correspond. La deuxième flèche descendante continue le cheminement de la première, elle va des logiciels vers les matériels et nous indique l'inscription du document encodé dans le support physique. Enfin, il y a la troisième flèche, celle qui remonte du matériel vers le logiciel : elle représente l'ensemble des procédés de réécriture possible des documents, tels que la modification, la conversion, la transformation ou encore la suppression. Fondamentalement, cette flèche montre la configuration du numérique qui rend inaccessible toute inscription faite sur un de ses supports, voire même les rend invisibles, si nous n'avions pas l'appui des interfaces logicielles - un terminal par exemple - pour y accéder. Se forme ainsi une boucle de communication entre logiciel et matériel pour écrire et réécrire dans l'espace numérique.

Conversion

La conversion d'un fichier ou d'un document est une phase cruciale de la forge. Cette phase consiste en la transposition d'informations d'un modèle de données à un autre modèle. Il s'agit le plus souvent d'informations structurées avec les contraintes d'un format de fichier, qui sont permutées avec un autre modèle de données afin d'obtenir un nouveau format de fichier. Les informations sont donc, d'une certaine façon, déplacées. Les textes et les métadonnées qui les accompagnent sont combinées afin d'obtenir un

nouvel agencement des mêmes informations. En d’autres termes, la conversion est une opération de réagencement de sources primaires selon des règles explicites.

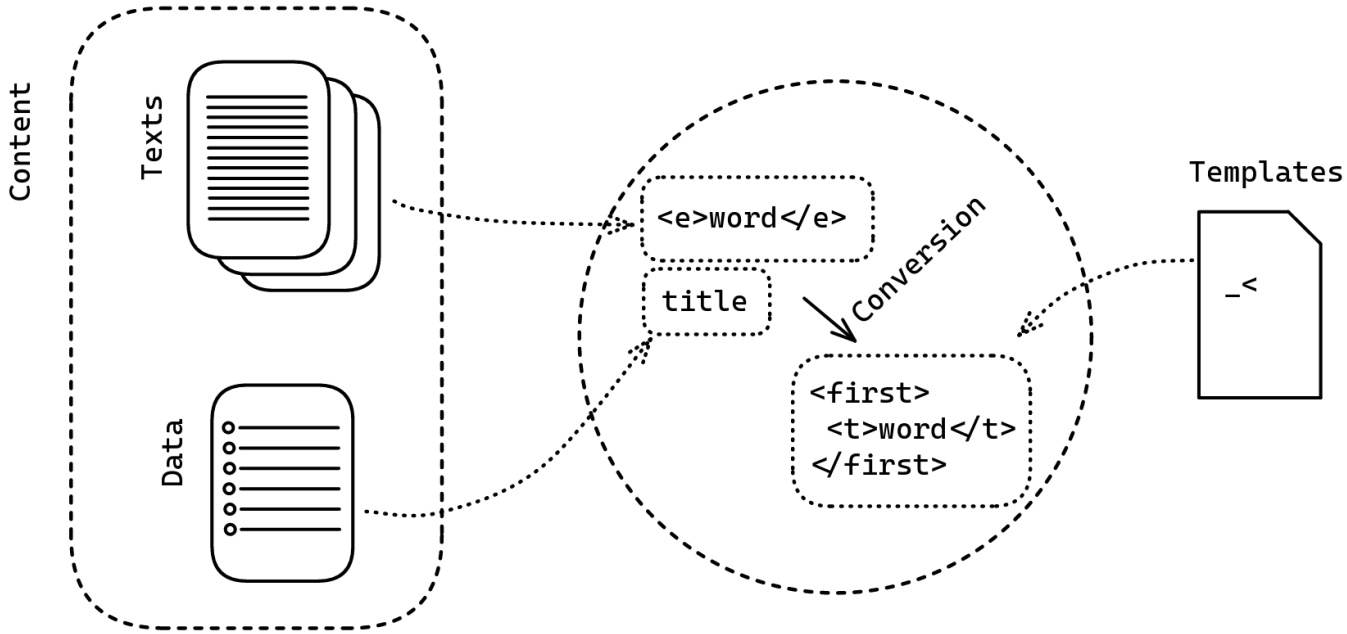

Figure 3. Schéma de la phase de conversion

La difficulté ne réside pas dans la modification du format primaire en un autre, mais dans la répartition ou le placement adéquat des informations, et dans la préservation de la richesse sémantique de la source primaire dans le fichier converti. La conversion est une forme de répartition des contenus depuis le document source vers un *nouveau* document qui nécessite l’établissement d’un modèle précis. Ce schéma de répartition ou cette feuille de conversion consiste à placer des contenus d’un fichier à un autre en modifiant la façon dont la structure est composée et dont la sémantique est exprimée. L’un des enjeux de la conversion consiste à vérifier que le processus n’a pas perdu d’informations. Ces vérifications sont importantes, particulièrement lorsqu’il est nécessaire de transformer la source primaire en plusieurs fichiers intermédiaires avant d’obtenir le fichier final.

Du côté du texte, l’avantage de travailler avec des langages de balisage, léger comme Markdown ou plus verbeux comme XML, est d’avoir un contrôle total sur les opérations de conversion à exécuter. Le document est un fichier texte (ou plein texte, ou texte brut), il n’y a donc aucune ambiguïté possible sur la façon dont les informations sont exprimées. Les technologies associées à ces conversions s’appuient sur des modèles d’organisation des données modulables selon les besoins, comme par exemple en éditant

complètement une feuille de transformation XSLT, ou en créant un programme de conversion depuis un langage de balisage léger comme Markdown.

La conversion est assurée par un ou plusieurs éléments logiciels, nous pouvons par exemple citer Pandoc qui permet d'appliquer un certain nombre de règles de conversion ainsi que d'utiliser des modèles de données pour acocmpagner cette opération. Les générateurs de site statique, évoqués plus longuement par la suite, intègrent un convertisseur permettant le passage d'un langage de balisage à un autre.

Transformation/production

La transformation est une opération (le plus souvent irréversible) qui consiste à passer d'un format à un autre. Il ne s'agit pas d'une conversion à proprement parler puisque le format obtenu après la transformation est souvent un fichier ou un ensemble de fichiers qui ne sont pas prévus pour être à nouveau convertis. Les artefacts résultant d'une transformation sont faits pour être lus par des logiciels (des fichiers HTML sont interprétés par un navigateur, des fichiers PDF sont lus par un lecteur PDF), ou pour être parsés par des programmes informatiques (des fichiers XML sont parcourus par des logiciels pour en extraire des informations).

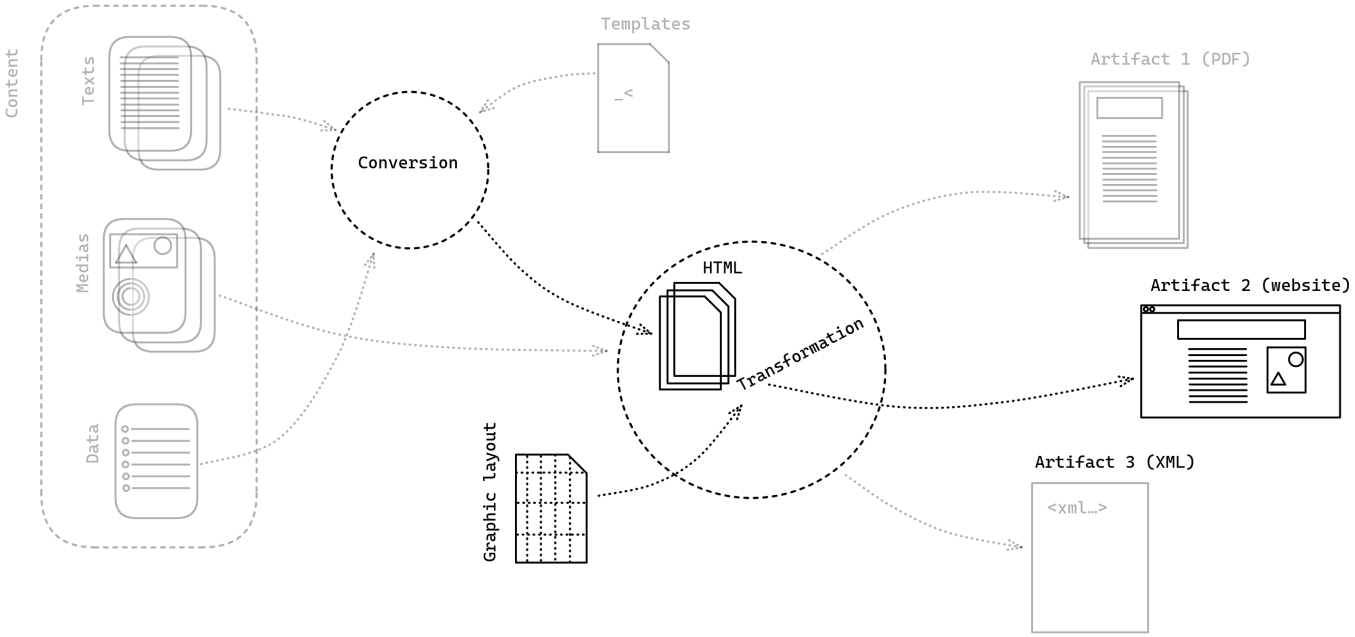

Figure 4. Schéma de la phase de transformation

Le cas typique est celui de la transformation d'un fichier HTML en un fichier PDF : une feuille de styles spécifique est appliquée à un ou plusieurs fichiers HTML, et l'ajout

d’un script permet d’émuler une mise en forme paginée pour une impression via un navigateur. La production des documents finaux est la dernière opération de la chaîne de publication, ce qui signifie que les artefacts ou les fichiers obtenus n’ont pas pour vocation d’être à nouveau transformés après cette étape – même si, comme nous le verrons plus loin, la pratique peut se révéler plus complexe. Une forge permet de produire les documents souhaités par conversion puis transformation d’une source primaire ou intermédiaire vers le format du ou des documents à produire. La particularité de cette étape est l’application de feuilles/grilles de styles aux documents convertis, afin de mettre en forme ou en page les différentes écritures. En appliquant une feuille de styles (graphique), ou une feuille de transformation (XSLT par exemple), les contenus structurés sont modifiés graphiquement ou sémantiquement pour être livrés ou être rendus accessibles.

La transformation/production peut être réalisée par un processeur, LaTeX, par exemple, contient un processus de transformation d’un fichier TEX en un fichier PDF. Autre exemple : Prince XML transforme un fichier HTML en fichier PDF.

Le point commun avec la conversion est qu’un certain nombre de règles doivent être appliquées pour que la transformation s’opère. Dans le cas d’un artefact qui est un site Web, il faut notamment :

- organiser/répartir les fichiers ;
- adapter les éléments de navigation en fonction de paramètres globaux ;
- appliquer une mise en forme à tout ou partie des contenus structurés ;
- définir des règles d’accès aux contenus ;
- etc.

Comme nous pouvons le voir ici, la transformation est l’étape qui fait le lien entre la conversion et la publication.

Publication

Cette étape ne rentre pas directement dans le spectre de ce rapport, toutefois nous devons présenter rapidement quelques contraintes qui peuvent avoir une influence sur la conversion et la transformation/production, voir même sur l’écriture.

En fonction de la destination – car il s’agit bien d’une destination, les fichiers vont être placés *physiquement* sur un serveur qui sera consulté par des agents humains ou non – des règles d’accès peuvent être appliquées. Il peut s’agir de limitations en terme

d’indexation par des moteurs de recherche pour le cas d’un site Web, ou de protections ajoutées sur des fichiers tels que les formats PDF ou EPUB. Ces éléments peuvent être pris en compte plus en amont dans la chaîne de publication, et ainsi être inscrits *avant* cette phase de publication.

Les questions de déploiement continu (Fauchié, 2021), c’est-à-dire le fait d’automatiser un processus afin qu’une étape déclenche les suivantes, peuvent également être définies dès le début de la chaîne et ainsi modifier les étapes de conversion et de transformation. Ainsi, chaque validation ou chaque enregistrement d’un ou plusieurs fichiers peut enclencher une série d’actions depuis la conversion jusqu’à la publication en passant par la transformation.

Enfin, un cahier des charges définissant le fonctionnement d’une chaîne de publication, et ainsi de la ou les forges qui la constituent, doit prendre en considération cette phase de publication.

PANORAMA DE 3 GÉNÉRATIONS DE FORGES

Nous avons détaillé ce que nous considérons comme étant des étapes génériques d’une chaîne de publication d’un projet éditorial. Depuis les débuts de l’informatique, plusieurs implémentations techniques ont répondu de près ou de loin à ce type de processus. Nous proposons ici une courte plongée dans plusieurs générations ou phases, permettant d’envisager des tendances, des évolutions et parfois des ruptures. Il s’agit le plus souvent d’évolutions lentes sous forme d’aller-retours. Nous considérons ici trois *moments* importants dans les constitutions des chaînes de publications : le balisage riche et les feuilles de transformations complexes avec XML au centre ; la construction de forges *monolithiques* pensées comme des solutions complètes et faciles d’accès, les CMS ; et enfin la mise à plat des données avec l’usage accru du format texte et plus particulièrement des langages de balisage dits légers.

Ces trois approches sont des phases successives de recherche de solutions adéquates pour des questionnements qui restent souvent les mêmes : comment convertir, transformer, organiser puis produire des documents riches dans des formats divers et en constante évolution ? Un regard critique sera porté sur ces solutions dans une prochaine partie, toutefois nous devons garder à l’esprit que les effets de mode sont nombreux. Le curseur se déplace entre les besoins des utilisateurs et des utilisatrices, les contraintes techniques et éditoriales, le confort des personnes qui développent et

maintiennent les forges, et l'adéquation entre fondements théoriques et réalités pratiques.

Les trois générations présentées ci-dessous sont apparues dans des temps successifs, mais 1) elles se croisent en réalité aujourd'hui et 2) elles vont vers une certaine simplification.

XML à tout faire

Si la version 1.0 de XML fait son apparition en 1998, ce langage de balisage a eu de nombreux prémices avec SGML puis la TEI.

XML est une série de recommandations permettant d'établir des schémas de données ainsi qu'un écosystème pour décrire très finement des informations. Un texte peut être facilement structuré grâce à des balises, et ensuite être converti dans divers formats grâce à des feuilles de conversion ou de transformation. L'objectif est de disposer d'un langage sémantique particulièrement riche qui permet d'exprimer des informations dans des contextes très variables. Le fait de pouvoir assez facilement décrire un document dans ses moindres détails — métadonnées, structure logique, description sémantique, identifiants, index, liens hypertextes, etc. — permet en effet d'envisager des usages très divers, du document imprimé au site Web avec moteur de recherche.

Utiliser XML comme format d'entrée d'une forge est un choix qui peut entraîner une certaine complexité :

- du côté de la création de la forge : les outils à disposition permettant de manipuler ou de convertir du XML sont limités en nombre, en usage et ;
- et de celui de la rédaction et de l'édition des contenus : les éditeurs de texte compatibles avec XML ne sont pas nombreux et sont souvent difficilement personnalisables.

En d'autres mots : XML est puissant mais complexe à manipuler.

CMS et bases de données relationnelles

Nous pouvons dire que XML, depuis la fin des années 1990, n'a pas toujours eu le niveau de reconnaissance qu'il méritait. Mal compris par les structures ou collectifs gérant les contenus, parfois boudé par les développeurs et les développeuses, d'autres solutions ont alors été imaginées.

Les CMS, ou systèmes de gestion de contenu, sont apparus non pas comme une alternative à l'approche XML, mais comme une solution complète permettant de répondre à des besoins très similaires. Des systèmes de gestion de contenu comme WordPress, Drupal ou Omeka-S répondent à trois principes :

1. proposer une interface graphique la plus simple (en apparence) possible ;
2. utiliser les bases de données relationnelles pour stocker les informations, et les agencer pour les rendre rapidement disponibles ;
3. penser une application ou un logiciel aussi complet que possible, et parfois extensible.

Les CMS offrent beaucoup d'avantages pour les personnes intervenant sur les contenus, et créent une distinction très nette entre cette catégorie d'utilisateurs et les personnes développant ces solutions (Kay, 1990). L'apparition des interfaces graphiques, leur usage intensif, et le recours à des bases de données relationnelles est aussi un manque à gagner en terme de pérennité, d'interopérabilité et de modularité.

L'approche de la JAMStack

En réaction aux CMS devenus des objets monolithiques de plus en plus difficiles à faire évoluer et trop complexes à migrer, une approche pas si nouvelle que cela a fait son apparition. Les générateurs de site statique ont introduit deux paradigmes très intéressants dans le cadre d'une démarche de publication :

1. la séparation entre la production d'artefacts et leur consultation (alors que bien souvent les CMS gèrent ces deux parties) ;
2. la perspective d'architectures découplées permettant de distinguer les différentes phases ou parties d'une chaîne.

La JAMStack est le résultat de l'utilisation de générateurs de site statique comme Jekyll, Hugo ou Gatsby : envisager une chaîne de publication comme l'assemblage de plusieurs couches logicielles. Un langage de balisage permet de structurer les contenus, le langage de programmation JavaScript permet d'ajouter de l'interaction côté client, l'appel à des APIs distingue une application de consultation de celle qui gère les données. Il faut ajouter à cela le fait de pouvoir versionner tout ou partie du projet puisque tout repose sur le format texte, les bases de données complexes sont abandonnées ou placées à la toute fin de la chaîne sans être le moyen de gérer les contenus.

Cette courte exploration de trois générations de forges successives sera complétée par plusieurs exemples ainsi qu'une synthèse des apports de chacune de ces approches.

EXPLORATION DE SOLUTIONS ACTUELLES

Ci-dessous, voici quelques solutions actuelles qui illustrent bien le panorama proposé.

Omeka S

Omeka S (pour Omeka Semantic) est un projet de Digital Scholar lancé en 2017 au *Roy Rosenzweig Center for History and New Media* et financé par plusieurs organisations. Omeka Semantic est un CMS nouvelle génération pensé pour créer des collections de documents interopérables qui peuvent être connectées avec d'autres ressources sur le Web. Cela est possible notamment grâce à l'intégration des technologies du Web sémantique dans le CMS, et grâce à la mise à disposition des données via une API REST. Sa construction s'appuie sur les technologies des bases de données relationnelles (MySQL, PHP) avec pour particularité d'indexer des *items* et les métadonnées qui y sont associées. Un changement de paradigme s'opère avec Omeka S : l'architecture traditionnelle des bases orientée "données" devient une architecture orientée "objet".

Quire

Quire est une chaîne de publication formée par un ensemble de logiciels et de programmes. Cette forge est développée et maintenue par le département édition numérique de la fondation/musée J. Paul Getty depuis 2017 pour les premiers prototypes. Quire est pensée pour produire plusieurs formats de sortie à partir d'une même source, tout en privilégiant la pérennité des données (en entrée et produites), la richesse des contenus, la lisibilité des formes produites et l'indépendance quant à son utilisation. D'abord lié au générateur de site statique Hugo, et Quire est désormais construit avec 11ty, adoptant un fonctionnement plus modulaire et l'approche de la JAMStack. Quire apporte une réponse aux enjeux de diffusion et de publication numérique et papier des collections muséales.

Sphinx

Sphinx fait partie de la première génération de générateur de site statique. Plus qu'un générateur de blog, comme le sont la plupart des générateurs de cette période, Sphinx

est initialement conçu pour générer des documentations techniques pour les projets développés avec le langage de programmation Python. Les évolutions proposées au fil des années, telles que les possibilités d'exporter les contenus dans divers formats, permettent à Sphinx de rester en lice sur le marché des générateurs de site statique et cela malgré les écarts qui se creusent avec la nouvelle génération de ce type d'application.

Svelte et SvelteKit

Svelte est une librairie de composants JavaScript, comme React, et SvelteKit est un *framework* pour créer des applications Web, très similaire à son homologue Next.js.

L'approche de Svelte est tout à fait novatrice vis-à-vis de ses concurrents : Svelte propose une amélioration des performances de construction des applications (lors de l'exécution du *build*) par l'utilisation d'un compilateur plutôt que de la traditionnelle différenciation dans le DOM virtuel. Cette composante nous permet d'observer une nouvelle évolution dans la lignée des générateurs de site statique, modifiant ainsi une dynamique qui paraissait s'essouffler depuis quelques temps.

RÉFÉRENCES

Fauchié, A. (2021). Déployer le livre. *Études du livre au XXIe siècle*. Université de Laval, Canada.

Fauchié, A., & Audin, Y. (2023). The Importance of Single Source Publishing in Scientific Publishing. *Digital Studies / Le Champ Numérique*. https://doi.org/10.16995/dscn.9655

Genette, G. (1982). *Palimpsestes. La Littérature Au Second Degré*. Paris: Seuil.

Hyde, A. (2021, août). *Single Source Publishing*.

Kay, A. (1990). User Interface: A Personal View. In *The art of human-computer interface design* (p. 191-207). Reading (Mass.) ; Addison-Wesley Pub. Co.

Kembellec, G. (2020). L'érudition numérique palimpseste. *Hermes, La Revue*, *n° 87*(2), 145-158.

Kittler, F. A. (2015). *Mode protégé*. Dijon: Les Presses du réel.

Mellet, M. (2020). Penser le palimpseste numérique. Le projet d'édition numérique collaborative de l'Anthologie palatine. *Captures : figures, théories et pratiques de l'imaginaire*, *5*(1). https://doi.org/10.7202/1073479ar

Vitali-Rosati, M. (2016). Qu'est-Ce Que l'éditorialisation ? *Sens Public*.

Vitali-Rosati, M. (2020). Qu’est-ce que l’écriture numérique ? *Corela. Cognition, représentation, langage*, (HS-33). https://doi.org/10.4000/corela.11759

CONTRAINTES

Tous les outils impliquent par essence des contraintes. Ce sont elles qui, en définissant les limites au-delà desquelles il devient complexe de se rendre, induisent les usages et les gestes les plus courants à réaliser pour atteindre un objectif précis avec un outil. Au-delà d'une attirance pour une technologie particulière, la connaissance de ces contraintes s'avère utile pour effectuer un choix au milieu de toute une panoplie d'outils réalisant sensiblement les mêmes tâches.

Naviguer entre les différentes forges mène à la rencontre des contraintes qu'elles transportent. Or, malgré le fait que certaines de ces contraintes soient très spécifiques, la nature de nombreuses d'entre elles est récurrente et réapparait d'une forge à l'autre. Néanmoins, toutes les forges ne sont pas contraintes par les mêmes limites. La création d'un outil, spécialisé ou non, a pour but la réalisation d'un besoin : il nécessite des choix d'optimisation vers le besoin en question, choix qui desserviront les autres réalisations possibles. Cette succession de choix à effectuer forme une arborescence des contraintes où chaque choix devient un nœud à partir duquel peut repartir une nouvelle branche. D'une forge à une autre, les embranchements de choix restent similaires : l'écosystème numérique, tissé par les ensembles de couches formelles – du *hardware* jusqu'au *software* –, présuppose déjà toute une variété de contraintes avant même les premières étapes d'élaboration d'une forge, et ces contraintes permettent de formuler les limites logicielles et matérielles de l'environnement dans lequel s'inscrit la forge. Les forges sont donc imbriquées dans une méta-catégorie de contraintes environnementales et des arborescences prédéfinies par ces dernières. Ainsi, en fonction des différentes arborescences de choix de fabrication, il devient possible de dresser une typologie des contraintes. À l'intérieur du processus de construction de la forge, nous dissocions plusieurs types de contraintes qu'elles soient d'usage, techniques ou encore sociales.

Tout d'abord, examinons la **contrainte d'usage**, que nous pouvons relier au design des outils. Un marteau peut tout à fait servir à enfoncer des clous ou à briser une tirelire, cependant, le même marteau sera difficilement utilisable pour enfoncer les piquets d'une clôture dans le sol. Ainsi, la contrainte d'usage se situe à l'interface entre l'outil et le geste d'utilisation, c'est-à-dire qu'elle incarne une distance relative entre deux pôles que sont la forme de l'objet et l'objectif de réalisation. L'oscillation de cette distance dépend notamment du niveau de maîtrise du geste technique de l'utilisateur. Plus les compétences de l'utilisateur sont élevées plus cette distance se réduit jusqu'à ce qu'il n'y ait quasiment plus de contraintes d'usage : l'outil peut aisément être détourné de son usage initial.

Cependant, la contrainte d'usage ne sera pas la seule qu'un utilisateur peut rencontrer. Même si cette dernière peut quasiment devenir inexistante, l'utilisateur se heurtera a une limite infranchissable : la **contrainte technique**. Les contraintes techniques regroupent tous les aspects relatifs à l'outil en lui-même, isolé de tout utilisateur. Elles concernent les caractéristiques techniques et matérielles de l'outil. Si nous reprenons l'exemple des marteaux, un maillet en bois ne peut pas être utilisé pour enfoncer des clous : il va se déformer voire se casser et devenir inutilisable. Les contraintes techniques sont propres à la nature et aux spécificités de l'outil et ne peuvent être dépassées sans risquer un incident (casse de l'outil, perte de données, etc.).

Le troisième type de contraintes est celui des **contraintes sociales**. Elles sont externes à l'outil. La contrainte sociale correspond à l'inscription politique de l'outil dans son écosystème. En effet, il s'agit des influences de l'écosystème social, donc des membres des communautés, sur l'outil. Cette contrainte provoque des incidences sur les développements de l'outil qui sont fonction des choix réalisés par les *communautés*. C'est-à-dire que la contrainte sociale agit en trois temps. Tout d'abord il y a le temps présent : un choix politique est fait, relatif à l'inscription dans une communauté précise. Ensuite le futur à moyen terme concerne les développements des outils. Il est possible de questionner les communautés concernées afin de connaître les feuilles de route des développements à venir : est-ce que ceux-ci correspondent aux besoins et aux usages souhaités ? Enfin il y a le temps long. Cette dernière temporalité est du ressort de la pérennité de l'outil. Certains outils peuvent très vite devenir obsolètes sans qu'aucun suivi ne leur soit accordé ; ils tomberont alors en désuétude et ne seront plus maintenus au risque de voir leur infrastructure devenir inopérante. Le cycle de vie d'un outil dépend de beaucoup de facteurs notamment financier, technologique, desiderata des utilisateurs, etc.

Cette brève typologie des contraintes révèle la complexité des enjeux et des rapports susceptibles de favoriser un choix d'outils en fonction des besoins et des usages souhaités par un utilisateur. Afin de préciser les contraintes qui touchent aux forges, nous présenterons dans la partie suivante certaines contraintes que nous avons recensées. Le choix a été fait de ne retenir que celles en rapport avec notre problématique d'éditorialisation pérenne de documents.

En conclusion, les éléments vus précédemment nous ont permis de faire un tour d'horizon des contraintes sur lequel nous nous appuierons lors de la présentation des fragments de forge de publication qui va suivre. De notre point de vue, les contraintes sont un

aspect primordial dans le choix d’utilisation d’une forge, quelle qu’elle soit. Cet enchevêtrement des contraintes permet, une fois ciblées, de préciser les ressources nécessaires à la réalisation d’un besoin particulier : celui pour lequel la forge est déployée.

COMPATIBILITÉ

Type : technique

La compatibilité est la capacité que deux éléments ou plus ont de coexister, d'être connectés ou d'être interdépendants, et ce sans générer de conflit. En d'autres termes la compatibilité est la possibilité de faire dialoguer deux éléments d'un système ou d'un processus.

La compatibilité est une contrainte, sans elle il est par exemple impossible de pouvoir lire un même fichier sur plusieurs systèmes d'exploitation. L'exemple typique désormais révolu est le fichier au format Microsoft Word qu'il était impossible d'ouvrir sur un système d'exploitation de type Linux : le fichier .doc n'était pas compatible avec tous les logiciels, et plus particulièrement sur certains systèmes d'exploitation.

Au sein d'un environnement numérique, la compatibilité concerne toujours deux technologies ou plus – matérielles et/ou logicielles – et peut se traduire par les capacités suivantes :

- interconnexion des technologies dans un même environnement sans interférence ;
- exécution d'une même instruction sans modification de programme alors que les technologies sont dans des environnements distincts (la rétrocompatibilité fait partie de cette catégorie).

Le compatibilité devient un enjeu fort dans des environnements complexes et enchevêtrés. Il ne s'agit plus de penser cet écosystème par organe bien séparés mais comme un tout où les différents éléments sont interconnectés. Modifier un des organes de l'écosystème signifie modifier les relations qu'ils entretiennent les uns avec les autres. Il faut alors veiller à ce que l'ensemble de l'environnement déployé reste stable au risque d'être rompu.

DESIGN

Type : usage

Le terme *design* ne renvoi pas dans ce rapport à la vaste discipline qu'il peut représenter. Il s'agit de regrouper les différentes propriétés auxquelles il peut référer. La contrainte *design* correspond à la forme donnée à l'outil, soit à ses caractéristiques matérielles. Ainsi, la dimension physique d'un logiciel contient différents paramètres comme les systèmes d'exploitation qui le supportent, les espaces de stockages, les capacités de calcul ou encore les moyens d'y accéder (voir la contrainte Design d'interface). Cette matérialité induit une utilisation dirigée de l'outil par le déploiement d'un écosystème adapté à son exécution. De manière générique, la plupart des contraintes découlent de la contrainte de design.

DESIGN D'INTERFACE

Type : usage

Le design d’interface peut être considéré comme la définition des éléments d’utilisation d’une application ou d’un logiciel. Il s’agit majoritairement de choix graphiques qui permettent d’appréhender des fonctions complexes, mais aussi de définition de scénarios d’usage pour définir les parcours d’utilisation.

L’interface d’un outil est devenue l’un des enjeux fondamentaux pour le développement et l’utilisation des logiciels. Les communautés d’utilisateurs favorisent des interfaces simples à prendre en main — ce qui est souvent sujet à débat, le terme de “simple” n’étant pas si évident à définir —, avec des instructions claires quand aux possibilités offertes. Par interface nous entendons une application permettant une communication entre des humains et des machines — en l’occurrence des ordinateurs —, qui peut être décomposée en plusieurs fonctions.

Une interface ne doit pas nécessairement être utilisée sans réflexion sur les fonctions appelées, ou sans une certaine friction entraînant des interrogations sur le fonctionnement général du logiciel utilisé. Certaines applications permettent justement d’engager les utilisateurs et les utilisatrices dans une attention à l’outil numérique utilisée, plutôt que des *usines à tout faire* où le fonctionnement devient complexe sans un réel apprentissage du logiciel.D’une certaine façon il en va de même pour les logiciels sans interface graphique autre qu’un terminal et des lignes de commande. L’enjeu n’est pas celui de la complexité mais du temps d’apprentissage. Si les courbes d’apprentissage sont trop raides, un outil n’aura pas une grande portée au-delà de la communauté qui l’a développé. À l’inverse, un outil qui *engage* les personnes qui y ont recours permet une utilisation plus maîtrisée, une transmission des connaissances acquises et une curiosité sur d’autres outils similaires. Lorsque l’utilisateur est pris dans une dynamique économique où chaque action doit être la source d’une production, l’apprentissage peut s’apparenter à une perte plutôt qu’à un investissement car il ne génère aucune ressource. Si un équilibre est trouvé entre l’acquisition de connaissances et la maîtrise de l’outil permettant un gain de temps, alors l’interface peut être considérée comme réussie.

DÉPENDANCE

Type : technique

Une dépendance technique correspond à ce dont un programme a besoin pour réaliser les tâches pour lesquelles il a été conçu. Il s'agit d'un lien de co-existence entre deux technologies, c'est un phénomène courant en informatique.

Une dépendance (technique) est une contrainte puisque cela signifie que pour qu'un programme informatique puisse fonctionner il faut qu'un autre programme soit disponible ou installé. Grâce à ce système de délégation, l'ensemble des fonctionnalités d'un logiciel ne sont pas à reprogrammer constamment, inutile de réécrire une fonctionnalité si elle a déjà été créée et qu'elle a fait ses preuves.

Avec l'avènement de la programmation orientée objet, beaucoup de bibliothèques de code ont vu le jour, comme autant de morceaux de programmes modulables. Ces bibliothèques peuvent être disponibles sur Internet, et ainsi être réutilisées pour accélérer le développement d'un programme. La difficulté réside dans le maintien de ces bibliothèques qui ne relèvent pas de l'environnement du logiciel final. En d'autres termes c'est à une autre communauté de s'occuper de cette maintenance, en sachant que le phénomène peut être pyramidal avec un effet domino : la dépendance en question peut elle-même dépendre d'autres bibliothèques de code, et ce de façon hiérarchique (une seule dépendance défaillante peut remettre en cause l'ensemble d'une forge).

L'accumulation de dépendances vis-à-vis de technologies diverses et variées peut devenir une source de fragilité de l'écosystème si une veille importante et rigoureuse n'est pas établie, d'autant plus dans la sphère informatique où l'obsolescence est un paramètre important. Certains choix sont faits pour construire des forges qui ont le moins de dépendances possibles, deux exemples peuvent être donnés ici :

- le générateur de sites statiques 11ty (Eleventy) offre une souplesse et une adaptation importantes grâce à un écosystème JavaScript foisonnant, ainsi de nombreuses bibliothèques JavaScript peuvent être ajoutées. La contrepartie est le nombre de dépendances (toutefois en diminution au fur et à mesure des évolutions de ce programme) qui sont nécessaires au bon fonctionnement de cette forge ;
- le générateur de sites statiques Hugo ne permet pas d'ajouter d'*extensions* en dehors des possibilités offertes par l'outil de *templating* (pour aller chercher et

afficher les données). En revanche Hugo contient toutes ses dépendances, il s’agit d’un programme informatique autosuffisant : un seul fichier binaire est nécessaire pour utiliser cette forge.

Attention il ne s’agit pas de réduire de façon absolue toutes dépendances mais plutôt de prendre conscience de cela comme une partie importante de la mise en place d’un projet d’édition avec le numérique.

DÉPLOIEMENT

Type : technique

Le déploiement est la mise à disposition d'une application ou d'un logiciel informatique en effectuant une suite d'actions techniques. Dans le cas d'une démarche éditoriale cela peut correspondre autant à la mise en place de la chaîne qu'à la production des artefacts (livres dans différents formats/formes).

Le déploiement d'une technologie peut devenir une contrainte si la complexité de sa mise en œuvre et sa maintenance sont trop complexes pour les personnes qui vont utiliser ces outils. Ainsi, cette étape cruciale pour l'utilisation d'une technologie doit pouvoir être réalisée facilement au risque de mettre en péril la chaîne éditoriale. Le déploiement étant d'une certaine façon l'aboutissement des étapes précédentes. Toutefois, de telles solutions complexes apportent *généralement* des bénéfices aux équipes sous la forme de fonctionnalités plus poussées ou plus précises pour certaines tâches.

L'enjeu du déploiement est de pouvoir rendre fluide, rapide et automatique des étapes techniques. La complexité du déploiement peut augmenter dans l'objectif de libérer du temps et de l'énergie pour les personnes impliquées — qu'elles aient un profil technique ou non. Dès lors, il devient pertinent d'évaluer les avantages procurés par ces solutions, accompagnés par leurs coûts (en ressources humaines ou pécuniaires) soit pour le recrutement d'une personne qualifiée au déploiement et à la maintenance, soit pour former les équipes et les faire monter en compétences face à un gain de temps et de ressources pour l'utilisation de technologies non personnalisables mais clefs en main.

Le déploiement continu est l'action plus spécifique de production et de publication d'artefacts éditoriaux au fil des modifications sur les contenus, sur la structure des données ou sur la mise en forme.

FORMAT

Type : technique

Le terme *format* est avant tout un terme technique, il délimite les caractéristiques d'un objet. Ces caractéristiques sont formulées par un certain nombres de données, d'instructions, ou de règles. L'objectif est de disposer d'un consensus pour dialoguer autour d'un objet ou de faire communiquer des processus qui traîtent ou qui produisent des formats.

Le format est une contrainte technique dans des environnements qui peuvent être très divers : formats d'objets physiques comme le papier, formats informatiques que nous connaissons par l'extension des fichiers sur nos ordinateurs, ou formats littéraires concernant l'agencement des mots et des phrases. Nous nous concentrons ici sur les contraintes techniques **et** informatiques. En fonction des nécessités d'un système d'exploitation, d'un programme informatique ou d'une plateforme en ligne, il faudra utiliser tel ou tel format. Un format qui n'est pas standard (ces caractéristiques doivent être décrites), qui n'est pas ouvert (il doit être possible de comprendre comment le format fonctionne) ou qui nécessite un environnement très spécifique pour être lu ou transformé va générer beaucoup d'obstacles pour son utilisation.

La contrainte du format est liée à d'autres contraintes comme la compatibilité (quel format peut être lu par quel programme ou logiciel ?), l'interopérabilité (est-ce que le format peut être utilisé de la même façon quel que soit l'environnement ?), la dépendance (de quoi un système a-t-il besoin pour traiter le format) et sa licence libre/open-source (est-ce que le format peut être lu, modifié, partagé ?).

Si le but du format est de constituer une série d'informations compréhensibles, utilisables et communicables, il reste une contrainte forte pour les chaînes de publication. Que ce soit en tant que format d'entrée, format pivot ou format de sortie, il déterminera le fonctionnement de la chaîne.

Enfin, le choix d'un format se fait en fonction de deux paramètres essentiels :

- le temps : est-ce que le format va devenir obsolète et ne sera plus reconnu par le ou les programmes de la forge ?

- la communauté : y a-t-il d’autres personnes en mesure de comprendre le format et d’apporter de l’aide (cas d’usage, solutions techniques, etc.) ?

INTEROPÉRABILITÉ

Type : technique

L’interopérabilité désigne une capacité d’interaction entre différents systèmes. Cette notion se dissocie de celle de compatibilité de par sa nature communicationnelle et ouverte, alors que la compatibilité ne concerne qu’un environnement fermé.

L’interopérabilité comporte deux principales contraintes :

- La première vise les systèmes fermés et non-intéroperables, qui, de fait, seront des vases clos non-communiquant et donc difficiles à intégrer dans un écosystème complexe et ouvert.
- La seconde contrainte concerne les protocoles à déployer pour qu’une chaîne soit interopérable. Cette caractéristique doit être envisagée dès le départ dans l’architecture de la chaîne de publication car elle nécessite d’intégrer des protocoles spécifiques.

À l’heure des données ouvertes et liées, l’architecture des silos de données isole complètement le propriétaire du reste du Web. Lier des données est une communication à double sens : il ne s’agit pas uniquement d’enrichir des corpus de données en ligne mais également d’enrichir et d’augmenter ses propres données.

LIBRE ET OPEN SOURCE

Type : sociale

Les communautés *open source* privilégient un accès libre et gratuit au code source d'une technologie. Cependant elles ne fournissent généralement aucun service d'accompagnement gratuit car elles doivent aussi s'inscrire dans un modèle économique leur permettant de subsister et de pérenniser leur structure. Les communautés autour du logiciel *libre* portent un projet politique qui vise d'abord l'émancipation avant le développement économique (marchand) d'un outil. Pour tenter de circonscrire la différence entre *open source* et *libre*, le premier répond clairement à des enjeux capitalistes de rendement quand le second porte un projet social qui ne souhaite pas forcément augmenter le profit des personnes qui le portent.

MATÉRIEL

Type : technique

Bien qu'elle paraisse évidente, la dimension matérielle d'une forge mérite toutefois de faire l'objet d'une contrainte. Dans la gestion d'un projet éditorial se pose toujours la question du matériel nécessaire à la réalisation des objectifs : quels sont les besoins, quels sont les outils à disposition, quels outils faut-il trouver pour compléter l'existant ? Les logiciels ne peuvent être décorellés de leur pendant tangible. Comme nous l'avons mentionné dans l'état de l'art, la partie *hardware* de la forge est le support de toute l'écriture numérique. D'ailleurs les logiciels ne sont compatibles qu'avec des configurations matérielles très précises. L'aspect matériel va donc conditionner nos artefacts à produire.

Au-delà de la question financière, le matériel peut devenir une contrainte lorsqu'il soulève les enjeux de compatibilité avec le reste du système, d'obsolescence, de prise en main et d'accessibilité, de dépendance et de propriété. Finalement, la quasi-totalité des contraintes que nous avons abordées dans le cadre du logiciel peuvent être transposables au cadre matériel d'une forge.

MULTILINGUISME

Type : technique

À l’échelle technologique, et plus spécifiquement à celle du Web, le multilinguisme concerne la capacité à traiter des langues différentes au sein d’une même plateforme. Actuellement, le Web est majoritairement anglophone, au détriment de toutes les autres langues qui ne sont que peu ou pas représentées. L’accès à des ressources bien indexées dans les autres langues s’avère complexe car ces ressources sont bien souvent maintenues par des communautés d’intérêt, minoritaires à l’échelle des utilisateurs du Web. Cependant, un certain engouement pour cette problématique émerge à grande échelle depuis quelques années. Les améliorations réalisées, techniques ou conceptuelles, notamment dans le champ du traitement automatique du langage, laissent espérer des avancées considérables dans la prise en charge du multilinguisme.

OBSOLESCENCE

Type : technique

La pérennité est une question primordiale qui se pose pour chaque édition d'un document numérique. La sphère numérique est régie par l'obsolescence, programmée ou induite par une évolution très rapide, qu'il s'agisse de la couche logicielle ou de la couche des supports matériels. Les langages de programmation, les formats des documents, et l'infrastructure technologique qui les supportent évoluent à une vitesse vertigineuse et soulèvent ainsi les questions de rétrocompatibilité et d'interopérabilité pour des documents à vocation pérenne. Dès lors un agencement des techniques utilisées est nécessaire si l'objectif envisagé est la publication d'un document à longue durée de vie, comme une archive.

PRISE EN MAIN

Type : usage

La prise en main d’une technologie est communément synonyme de l’apprentissage technique et fonctionnel de cet outil. Au-delà de la question du design, traitée comme une contrainte à part entière dans une autre fiche, la prise en main sous-entend l’apprentissage et la compréhension des fonctionnalités permises par l’outil et surtout leurs limites. En fonction des technologies et de leur complexité, cet apprentissage peut s’avérer plus ou moins long.

La prise en main d’une technologie peut devenir une contrainte si elle est trop complexe car elle s’avère alors gourmande en terme de ressources humaines et ressources financières. Investir une technologie nécessite ainsi une réflexion autour des intérêts qu’elle revêt vis-à-vis de son implémentation dans une forge, notamment pour les enjeux de compatibilité, de pérennité et d’interopérabilité.

PROPRIÉTAIRE

Type : sociale

Est qualifié de “propriétaire” un logiciel ou un programme informatique dont les usages sont fortement limités : licence d’utilisation qui limite les usages, impossibilité de lire le code source, ou interdiction ou impossibilité de modifier le fonctionnement initial du programme. Certaines structures — majoritairement des sociétés ou des entreprises — proposent des services payants pour l’accès, le droit d’utilisation, la mise à jour et l’évolution d’un outil. Les raisons de ces limitations ou de ces restrictions sont majoritairement économiques : en *fermant* l’accès au fonctionnement du programme l’entreprise qui le distribue espère pouvoir conserver son monopole (limiter la diffusion non maîtrisée et garder la recette secrète).

Un programme propriétaire implique parfois des facilités de mise en place et de déploiement côté utilisateur : les services dits en SaaS (pour Software As A Service) sont des exemples de logiciels/programmes propriétaires contemporains, ils sont déjà configurés et il suffit juste de se connecter en ligne pour accéder aux différentes fonctionnalités proposées. Toutefois le code source de ce type de technologie n’est pas rendu disponible et s’il advient que l’entreprise est obligée de fermer définitivement, le code source n’est plus maintenu et l’utilisateur ne peut plus s’en servir — quand bien même il aurait les compétences pour modifier le code.

Le caractère *propriétaire* d’un programme ou d’une forge pose des problèmes en terme de compatibilité, d’adaptabilité et d’obsolescence.

RÉTROCOMPATIBILITÉ

Type : technique

La rétrocompatibilité est à différencier de la compatibilité. La première désigne la capacité d'un matériel/logiciel à prendre en charge les commandes des générations de matériels et logiciels antérieures, il s'agit d'une composante de la compatibilité, alors que la deuxième indique plus largement la capacité d'agencement entre différentes technologies.

Les évolutions dans le champ du numérique sont nombreuses et l'intervalle entre chacune est de plus en plus rapide. Pour chaque technologie, la question de la rétrocompatibilité est posée : est-ce que la version actuelle prend en charge les instructions de la version précédente de la même technologie ? S'il y a rétrocompatibilité, c'est-à-dire que la même utilisation peut être faite quelque soit la version, il n'y a pas ou peu de contrainte. S'il n'y a pas de rétrocompatibilité cela sous-entend deux principales possibilités pour contourner la contrainte qu'imposerait une mise à jour de l'outil vers une version plus récente :

- une révision totale de la structure ;
- ne pas effectuer la mise à jour et rester sur avec la version précédente.

Les deux choix impliquent des politiques de gestion de la structure différentes. Le premier à un coût conséquent : ces changements sont majeurs et modifier une structure opérationnelle est une étape délicate mobilisant beaucoup de ressources. Le second choix renvoie au paradigme de l'obsolescence. S'il est difficile d'évaluer une moyenne selon les différents outils, une version d'un outil n'est maintenue tout au plus que quelques années, sauf exception comme par exemple avec le langage XML dont la dernière version stable 1.1 date de 2004 ou le langage HTML qui est rétrocompatible par conception. On peut qualifier cette période de durée de vie d'une technologie. Cette durée de vie correspond à la période de compatibilité d'une technologie dans une infrastructure, c'est-à-dire la période pendant laquelle il n'y pas de mise en péril de la stabilité de cette dernière. L'arrivée à la fin de cette période signifie devoir revenir au premier choix sous peine de générer une fracture dans la chaîne technologique.

TAILLE DE LA COMMUNAUTÉ

Type : sociale

Qu'il s'agisse d'un programme/logiciel propriétaire ou libre/open-source, la taille d'une communauté qui gravite autour de cet outil mis à disposition des usagers peut représenter une contrainte.

Ce que nous appelons "communauté" correspond aux personnes qui développent, maintiennent et utilisent une forge. Il peut s'agir des personnes qui ont créé l'outil, de celles qui participent à son évolution mais également de celles qui en ont tout simplement besoin. Dans le domaine du logiciel libre – et également open-source à un moindre niveau – cette notion de communauté est essentielle, puisque c'est grâce à elle qu'un outil peut être maintenu et évoluer.

Tout logiciel ou programme informatique repose sur du code qu'il est nécessaire de documenter voire de mettre à jour. Que ce soit la correction d'un problème – un *bug* –, l'amélioration d'une composant ou la création d'une nouvelle fonctionnalité, la plupart du temps les objets numériques nécessitent des mises à jour. Par ailleurs, une communauté offre aussi des opportunités de mutualisation humaine et financière pour faire vivre l'outil qui réunit cette communauté.

Une faible communauté ne permet pas toujours de disposer de forces suffisantes pour maintenir un outil, à l'inverse une communauté trop importante peut générer des demandes divergentes ou des tensions humaines. Cela dépend autant de la complexité de l'outil, de la popularité du ou des langages informatiques utilisés ou des profils des personnes impliquées – de *simples* usagers ne suffisent pas pour maintenir un outil. Certaines forges nécessitent de disposer d'un bassin d'utilisateurs et d'utilisatrices conséquent pour pouvoir être maintenues : les scénarios d'usage doivent être décrits, le code et les fonctionnalité doivent être testés, etc. En revanche d'autres programmes fonctionnent très bien depuis de nombreuses années avec une communauté restreinte mais régulièrement active.

LES FORGES EN DÉTAIL

Nous analysons un certain nombre de *forges* selon une sélection qui résonne avec notre problématique générale, et qui répond aux critères exposés ci-dessous. La grille d'analyse également détaillée ci-après constitue la structure de ces analyses. Nous devons préciser que ces *forges* sont constituées de technologies, et qu'une forge est censée correspondre à une chaîne de publication complète, un convertisseur, un processeur ou tout autre programme/logiciel pensé pour produire des documents structurés. En d'autres termes, ces forges peuvent être tout ou partie d'une chaîne de publication et répondent, de manière tant ciblée que plus large, aux besoins relatifs à la réalisation de ce type de processus technique permettant de produire des objets éditoriaux. Aussi, les forges présentées ne sont qu'un petit échantillon des solutions existantes dans la sphère de l'édition numérique. Il s'agit de s'appuyer sur des technologies qui permettent de dégager certaines fonctionnalités et/ou des contraintes précises, illustrant ainsi le chapitre précédent.

CRITÈRES DE SÉLECTION

Comme présentés dans l'introduction, les critères de sélection sont les suivants :

- **nature/objet** : l'objectif ici est la production d'un objet éditorial, les forges sélectionnées doivent avoir comme objectif de répondre à ces enjeux. Les convertisseurs de document, les processeurs ou les générateurs de documents entrent dans ce cadre ;
- **libre/*open source*** : les outils, programmes ou logiciels doivent être mis à disposition sous une licence libre ou à défaut sous une licence dite *open source* {référence nécessaire}. Il s'agit d'une condition nécessaire pour que ces outils puissent être utilisés, intégrés dans d'autres processus voire modifiés et redistribués ;
- **standards** : ce critère est plus compliqué à préciser exactement, mais il s'agit ici de faire appel à des langages qui sont créés et maintenus par la communauté, de façon ouverte et documentée ;
- **modularité** : ce critère découle des précédents, puisque la modularité — ou le fait de pouvoir agencer et interchanger plusieurs programmes ou logiciels constituant une chaîne de publication — dépend de l'usage de standards qui permettent une interopérabilité.

Les forges retenues répondent à au moins trois de ces critères (les trois premiers), la modularité faisant office de complément.

GRILLE D’ANALYSE

La grille d’analyse utilisée pour structurer les fiches se compose des éléments suivants :

- **descriptions courte** : présentation succincte des objectifs de la forge et notre intérêt pour celle-ci ;
- **description longue** : description du fonctionnement, historique de création et d’évolution, acteurs impliqués, évocation des dépendances et de l’environnement technique ;
- **formats des fichiers en entrée** : description des formats de contenu gérés par la forge ;
- **formats des fichiers en sortie** : description des formats des artefacts produits par la forge ;
- **usages et exemples** : domaines d’usage et exemples de projets et de cas d’utilisation ;
- **technologies utilisées** : description des technologies utilisées pour le fonctionnement de l’outil (aussi appelées *dépendances*) ;
- **interfaçages** : description des procédés permettant aux interfaces graphiques l’édition de contenus, l’organisation des informations ou la modification de paramètres ;
- **contraintes** : l’ensemble des règles d’utilisation comme les formats de balisage et leurs spécificités, ou encore certaines limites comme des environnements informatiques spécifiques.

11TY

DESCRIPTION LONGUE

11ty (ou Eleventy) est un générateur de sites statiques créé par Zach Leatherman écrit en JavaScript. 11ty fait partie de la génération de générateurs de sites statiques pensés pour faire autre chose que des sites web de type vitrine ou blog. C'est un outil flexible basé sur JavaScript mais ce n'est pas un *framework* JavaScript.

11ty reprend des fonctionnalités déjà utilisées dans d'autres générateurs comme Jekyll pour l'organisation des contenus en dossiers, ou l'extensibilité comme Jekyll ou Gatsby. Même si Eleventy n'atteint pas les temps de production très courts de Hugo (quelques millisecondes pour produire plusieurs centaines de pages), il est bien plus rapide que d'autres générateurs comme Jekyll.

Le succès d'Eleventy a été très rapide, pour deux raisons principales :

- l'environnement JavaScript très répandu permet une adoption facile ainsi qu'une large communautés de contributeurs et de contributrices ;
- Zach Leatherman a une façon de gérer ce projet très ouverte, bienveillante, et chaleureuse. Les personnes qui commencent à contribuer à 11ty découvrent une communauté accueillante.

Il faut noter que depuis 2020 Netlify finance Eleventy, notamment en payant Zach Leatherman à temps plein pour travailler sur le projet. L'intérêt pour Eleventy est de développer un certain nombre d'outils et de services autour de 11ty.

LANGAGE DE PROGRAMMATION

11ty est développé en JavaScript.

L'enjeu pour les générateurs de sites statiques est le langage de *templating*, la plupart ne prennent en compte qu'un seul langage pour la modélisation des contenus. La particularité d'Eleventy est de prendre en compte une grande variété de langages de *templating*, et donc de pouvoir être utilisé dans des situations très diverses sans imposer une façon d'écrire les modèles de contenus ou de données.

INTERFAÇAGE

Plusieurs *Headless CMS* peuvent être connectés à 11ty, tels que CloudCannon ou Netlify CMS. En raison du langage de programmation utilisé, il est relativement simple de développer des composants graphiques permettant un *interfaçage* avec 11ty.

EXEMPLES D'UTILISATION

Comme la plupart des générateurs de site statique, 11ty sert à produire des sites web de type vitrine (quelques pages de présentation) ou blog (une série de billets antéchronologique). En pratique 11ty se révèle à la fois souple et extensible, ce qui en fait un candidat parfait pour produire autant des sites web dits classiques que des applications web.

Nous pouvons citer ici l'utilisation de 11ty dans la chaîne de publication Quire, utilisation récente qui vient remplacer un autre générateur de site statique, Hugo. L'intérêt pour l'équipe du Getty qui développe Quire, c'est de pouvoir bénéficier d'un environnement commun pour la majorité des briques : JavaScript.

CONTRAINTES

11ty apporte quelques contraintes, certaines sont liées à la majeure partie des générateurs de site statique, d'autres sont liées à l'environnement JavaScript en soi.

Ce qui peut être considéré comme une contrainte majeure est l'environnement JavaScript. 11ty étant extensible, cela signifie que des bibliothèques JavaScript peuvent utilisées. 11ty a déjà un certain nombre de dépendances, mais l'usage de bibliothèques externes — et donc développées par d'autres personnes — ajoute une contrainte de dépendances. Pour l'exprimer autrement, il n'est possible d'utiliser 11ty que grâce à un certain nombre d'autres programmes, cette liste pouvant s'étendre très vite lorsqu'on utilise d'autres extensions et que celles-ci dépendent elles-mêmes d'autres bibliothèques, etc. Ce problème, lié à l'environnement JavaScript et Node.js, est déjà bien identifié, et est la contrepartie de l'extensibilité et de la souplesse de cet outil.

ASTRO

DESCRIPTION LONGUE

Astro est un générateur de site statique nouvelle génération. Le principe de fonctionnement novateur d'Astro lui permet de générer des sites très légers. Lors de l'exécution du *build* (la phase de production des fichiers depuis les sources), Astro interprète le JavaScript et le transforme directement en HTML pour limiter les calculs du côté du client. Ainsi les applications statiques sont bien plus légères et rapides que celles issues des générateurs de l'ancienne génération. Ce principe, Astro le nomme « hydratation partielle ». Les développeurs choisissent explicitement les composants qui doivent être dynamiques sur la page affichée sur le Web et tous les autres sont interprétés en HTML.

Un autre atout d'Astro est qu'il rend possible l'implémentation des *frameworks* les plus répandus pour le développement de composants. Les développeurs peuvent créer des composants avec le *framework* qu'ils utilisent habituellement, intégrant même la technologie de Svelt que nous détaillons dans une autre fiche.

Astro peut être comparé avec 11ty sur deux différences majeures :

- Astro prend en charge des *frameworks* ou des langages de *templating* plus complexes, permettant d'imaginer des applications ;
- Astro propose l'hydratation partielle contrairement à 11ty qui se situe plutôt sur la génération de fichiers HTML.

LANGAGE DE PROGRAMMATION

Astro repose sur l'environnement Node.js et est écrit avec le langage de programmation JavaScript. Il est possible d'utiliser la plupart des *frameworks* modernes basés sur Javascript pour créer des composants de l'interface graphique. Une bibliothèque met à disposition des modules pour intégrer différentes technologies comme Pandoc dans le générateur.

INTERFAÇAGE

Un module d'interfaçage avec Netlify est disponible dans la bibliothèque de modules. L'utilisateur peut ainsi utiliser un tableau de bord avec Netlify pour gérer le contenu à déployer en ligne.

EXEMPLES D'UTILISATION

Les exemples partagés sur le site d'Astro sont principalement des sites vitrine, bien que ses usages soient très larges.

CONTRAINTES

La contrainte principale provient certainement du large environnement qu'est Node.js puisqu'il n'est pas aisé de le prendre en main lorsqu'on est néophyte. Un usage standard d'Astro sera toutefois relativement simple à déployer : en effet, Astro devance les questions que l'on peut se poser lors des premiers essais en mettant à disposition des utilisateurs une documentation complète, des *code sandbox* pour visualiser des échantillons de structure du code et des thèmes basiques préfabriqués pour créer des premiers projets. Toutefois, l'intégration et la gestion de certains outils externes, comme la prise en charge de Zotero et du BibTeX pour intégrer des références bibliographiques dans le site web, demandera une utilisation plus fine et avancée d'Astro afin d'arriver à ses fins.

EDITORIA

DESCRIPTION LONGUE

Editoria est une chaîne de publication permettant de produire plusieurs formats de livre à partir d'une source unique, donc en appliquant le principe du *Single Source Publishing*.

Editoria est développé par la fondation Coko : fondée en 2015 par Adam Hyde, *Coko Foundation* a pour vocation de fédérer une communauté internationale autour de la création de chaînes de publication reposant principalement sur les technologies du Web. Les valeurs promues par cette Fondation sont axées sur la communauté et la collaboration, le développement d'outils *open source*/libre et la construction de chaînes de publication modulables et adaptées aux besoins des différents acteurs de la chaîne éditoriale. L'exemple de la plateforme Kotahi montre de manière explicite la construction d'une chaîne interopérable et centralisée autour d'une seule plateforme dans laquelle peuvent intervenir tous les rôles nécessaires à l'aboutissement d'une édition de type revue scientifique. Dans le cas de Kothai, le format fournit en entrée est le docx, puis il sera transformé par la plateforme en XML JATS pour servir, selon le principe de *single source publishing* les différentes sorties souhaitées.

Editoria offre une interface d'écriture facile à prendre en main, et permet à plusieurs personnes d'intervenir sur les différentes parties d'un livre – mais simultanément. Les sources sont au format HTML, mais la seule façon d'y accéder reste l'interface graphique qui est une application web. Les formats de sortie sont le HTML (site web structuré en plusieurs parties/chapitres), le PDF (via Paged.js qui est un autre composant développé par Coko) et l'EPUB.

LANGAGE DE PROGRAMMATION

Cette application est principalement développée en JavaScript.

INTERFAÇAGE

Editoria propose un outil tout-en-un avec une interface graphique déjà intégrée, il n'est donc pas nécessaire d'envisager un interfaçage.

EXEMPLES D'UTILISATION

La communauté qui gravite autour de la Fondation Coko compte parmi ses membres des universités situées sous toutes les latitudes du Globe. Quelques exemples de partenaires dont la chaîne de publication a été construite avec l'aide des outils de la Fondation Coko :

- https://www.micropublication.org/
- https://www.digital-science.com/
- https://head-publishing.ch/
- https://online.ucpress.edu/collabra

Dans le cas précis d'Editoria, le projet BookSprints est un bon exemple d'utilisation. Tous les livres produits dans le cadre de BookSprints utilisent Editoria. Cela permet à une communauté de concevoir, écrire et éditer un projet à plusieurs.

CONTRAINTES

Les deux contraintes qui apparaissent sont l'installation et le déploiement de l'application/chaîne de publication Editoria. Editoria doit être installée sur un serveur avec un certain nombre de prérequis techniques. Une telle technologie n'est pas à la portée d'un néophyte.

ELECTRIC BOOK

DESCRIPTION LONGUE

Electric Book est une chaîne de publication complète proposée par la structure Electric Book Works, basée sur le générateur de site statique Jekyll. Electric Book permet d'appliquer le principe du *Single Source Publishing* : à partir d'une source unique, ici des fichiers au format Markdown avec des entêtes au format YAML, il est possible de produire des artefacts aux formats web (site web ou application), PDF et EPUB. Cette chaîne est composée des éléments suivants :

- The Electric Book template : un modèle pour structurer les textes et générer des formats HTML pivots via Jekyll permettant ensuite de produire les artefacts finaux ;
- The Electric Book Manager : une interface graphique sous la forme d'une application web.

LANGAGE DE PROGRAMMATION

Electric Book est principalement basé sur le générateur de site statique Jekyll, un générateur de site statique très populaire au milieu des années 2010.

INTERFAÇAGE

Electric Book propose un composant permettant l'édition des contenus via une interface graphique (application web). Le déploiement est relativement simple puisque les fichiers produits sont des fichiers *statiques* principalement HTML et que Jekyll est bien pris en charge dans les environnements de déploiement classiques.

EXEMPLES D'UTILISATION

Sur le site d'Electric Book Works plusieurs exemples sont présentés : https://electricbookworks.com/work/

CONTRAINTES

Electric Book impose une organisation particulière des contenus, ainsi qu’un formatage des formes produites.

HUGO

DESCRIPTION LONGUE

Hugo est un générateur de site statique créé par Steve Francia en 2013 et qui se distingue par sa rapidité de production. Si Hugo reste le générateur de site statique le plus rapide, il offre plusieurs autres fonctionnalités intéressantes dans le cas de projet d'édition :

- l'absence de dépendances : seul un fichier binaire est nécessaire pour utiliser Hugo ;
- la multimodalité : Hugo peut convertir une même source dans plusieurs formats de sortie différents, ce qui en fait un très bon candidat pour l'application du principe de Single Source Publishing ;
- les *shortcodes* : comme d'autres générateurs de site statique, Hugo permet d'utiliser un balisage spécifique. Cette injection de codes personnalisés peut être très pratique pour pallier les limites de Markdown et prévoir des comportements différents entre différents formats de sortie.

Si Hugo est d'abord pensé pour construire des sites web relativement classiques (sites vitrines, blogs, documentation), il est possible d'imaginer des usages dans le domaine de la publication.

LANGAGE DE PROGRAMMATION

Hugo est développé en Go, mais son utilisation ne nécessite en aucun cas l'apprentissage de ce langage. Hugo utilise un langage de *templating* spécifique, facile à prendre en main pour construire des objets éditoriaux.

INTERFAÇAGE

En raison de la popularité de Hugo, il existe de nombreuses possibilités pour déployer ou coupler le générateur avec des interfaces d'écriture et d'édition de contenu.

EXEMPLES D’UTILISATION

Les sites utilisant Hugo sont nombreux, nous pouvons citer deux exemples emblématiques :

- pendant un temps la chaîne de publication Quire utilisait Hugo pour la conversion et l’organisation des fichiers, avant de lui préférer 11ty ;
- le recueil de textes Le Novendécaméron utilise Hugo pour la production du site web et de la version imprimable (à venir).

CONTRAINTES

Hugo a un fonctionnement relativement rigide, obligeant à adopter une certaine organisation de fichiers, ainsi que des déclarations de configuration particulières. Par ailleurs, Hugo ne permet d’utiliser des extensions de façon aussi souple que 11ty par exemple.

JEKYLL

DESCRIPTION LONGUE

Jekyll est un générateur de site statique écrit dans le langage de programmation Ruby. Son fonctionnement est classique, il assure la conversion d'un langage de balisage léger vers HTML, et il répartit les pages à partir d'une organisation de fichiers et de dossiers, et d'un fichier de configuration. Jekyll est extensible, il est possible de créer des *plugins* écrits en Ruby. Il existe un vaste écosystème de programmes permettant d'étendre les usages de Jekyll — par exemple pour une gestion plus fine de la typographie ou pour l'affichage de références bibliographiques.

Jekyll est important dans le paysage de la JAMStack (voir glossaire), il est le générateur de site statique le plus utilisé et qui a donné lieu à plusieurs autres solutions techniques intéressantes. Créé par Tom Preston-Werner en 2008, Jekyll est d'abord pensé comme un outil pour générer et gérer un blog. Bientôt intégré dans GitHub Pages, la mini-plateforme de publication de sites web de GitHub, Jekyll va connaître un succès très important.

Jekyll présente plusieurs limites, et notamment le temps de génération. Plusieurs générateurs de site statique ont, semble-t-il, remplacé ce pionnier :

- Hugo (voir la fiche) pour sa rapidité ;
- 11ty (voir la fiche) pour sa souplesse.

Langage de programmation

Jekyll est écrit en Ruby, ce qui permet à une communauté assez importante de contribuer au programme ou de proposer des extensions. Moins populaire que d'autres langages, Ruby a probablement bénéficié de cette visibilité offerte par le générateur de sites statiques.

Interfaçage

Jekyll a été l'un des générateurs de site statique les plus utilisés, probablement jusqu'aux années 2017-2018. Ainsi plusieurs services en ligne proposent un interfaçage graphique — souvent via Git — comme Forestry.io, CloudCannon ou Netlify CMS.

La structure relativement figée des fichiers et des dossiers permet d'imaginer des interfaces graphiques qui viennent se brancher sur Jekyll. Une liste exhaustive ne peut pas être proposée ici, mais les solutions sont nombreuses.

Exemples d'utilisation

Dés le début des années 2010, beaucoup de sites web de petite ou moyenne taille sont passés de WordPress à Jekyll. Les développeurs et les développeuses derrière l'administration technique de ces sites ont préféré un confort de travail à une édition en mode WYSIWYG. Ce qui a ensuite été nommé "DX" — pour Developer eXperience — a été un moteur de changement. Les sites fabriqués avec Jekyll sont donc nombreux, mais voici quelques exemples de projets de plus grande envergure ou qui ont une véritable démarche éditoriale :

- le site web *vitrine* de CloudCannon : https://cloudcannon.com
- le catalogue *Les sculptures de la villa romaine de Chiragan* : https://villachiragan.saintraymond.toulouse.fr

Contraintes

La contrainte principale est probablement celle liée à Ruby, l'installation et la mise à jour de Jekyll n'est pas toujours aisée — contrairement à Hugo (lien vers la fiche) par exemple qui ne nécessite qu'un fichier binaire.

L'autre contrainte observée rapidement par la communauté est le temps de génération ou de *build* pour des objets/sites web qui dépassent la dizaine de pages. Le temps de génération est très long, atteignant plusieurs minutes pour une quantité importante de contenu (nombre de pages) et selon la complexité des modèles/*templates*.

MÉTOPES

DESCRIPTION LONGUE

Le projet Métopes, actuellement dirigé par Dominique Roux, est porté par l'infrastructure de recherche Métopes et s'articule autour d'un réseau de partenaires composé de l'Université de Caen, du CNRS-InSHS, du Certic et de la MSRH. L'objectif de ce projet est de créer une chaîne de publication dédiée à l'édition scientifique à partir d'une seule source reposant sur les technologies du langage de balisage XML (en suivant le principe du *Single Source Publishing*). Les développements ont été concentrés sur les documents issus du logiciel Microsoft Word (autour du format .docx) qui est l'un des outils de traitement de texte le plus utilisé (en général et plus particulièrement dans le domaine académique) pour la rédaction de documents. La chaîne Métopes repose sur les transformations du XML grâce aux scripts XSL : l'outil permet de « baliser » le texte depuis l'interface de Microsoft Word puis applique une transformation XSLT au document afin d'obtenir un document source interopérable et à la sémantique riche (XML TEI). Ce document pourra par la suite faire l'objet de nouvelles transformations en fonction des formats de diffusion désirés : vers le logiciel InDesign de la suite Adobe pour une mise en page *print*, en XML/TEI vers la plateforme OpenEdition, en HTML ou encore vers d'autres processeurs.

INTERFAÇAGE

L'outil de Métopes est accessible *dans* Microsoft Word via l'usage de macros et plus spécifiquement une barre d'outils dans le traitement de texte. L'objectif premier de Métopes est de fournir des documents transformables et adaptables aux différents usages souhaités par les utilisateurs.

EXEMPLES D'UTILISATION

Métopes est employé pour produire des documents XML qui sont ensuite diffusés sur des plateformes comme OpenEdition ou le portail Cairn en France. Du fait de son implémentation dans le logiciel Microsoft Word, Métopes est devenue une chaîne largement utilisée pour les publications scientifiques par des revues.

CONTRAINTES

Malgré le développement de plugins installables dans plusieurs logiciels (LibreOffice Writer, OpenOffice Writer, Microsoft Word), la contrainte majeure de cette forge est qu’elle repose principalement sur l’utilisation de Microsoft Word et sur le format docx. Ainsi, les éditeurs qui n’utilisent pas ce logiciel ne peuvent pas utiliser ces outils pour construire leur chaîne éditoriale.

Afin de pallier les éventuelles difficultés de prise en main de cette solution, la structure porteuse de l’outil dispense des formations aux différents publics amenés à intégrer Métopes à leur chaîne de publication.

OMEKA S

DESCRIPTION LONGUE

Omeka S est un projet de Digital Scholar lancé en 2017 au Roy Rosenzweig Center for History and New Media avec le financement de plusieurs organisations. Les caractéristiques principales d'Omeka S viennent compléter le premier CMS développé par la même institution *Omeka Classic* : gestion d'un nombre illimité de sites web à partir d'une seule interface, conception de la base de données selon les principes du *linked open data* et interopérabilité avec d'autres systèmes notamment grâce à une API REST intégrée dans le logiciel. Sa construction s'appuie sur les technologies des bases de données relationnelles (MySQL, PHP) avec pour particularité d'indexer des *items* et les métadonnées qui y sont associées. Un changement de paradigme s'opère avec Omeka S, l'architecture traditionnelle des bases orientée "données" devient une architecture orientée "objet".

LANGAGE DE PROGRAMMATION

Omeka S repose sur les technologies MySQL et PHP pour la gestion des données.

INTERFAÇAGE

Omeka S est un CMS requérant une base de données dans laquelle stocker et éditer les données. L'environnement en lui-même est complet et ne nécessite pas un interfaçage avec d'autres technologies. Cependant, la visualisation des données est l'un des points faibles du CMS. Il est toutefois possible de contourner cet obstacle en utilisant l'API pour récupérer les données indexées dans Omeka S et générer des visualisations à partir d'autres outils comme D3JS ou Gephi.

EXEMPLES D'UTILISATION

Omeka S est principalement destiné aux institutions, notamment pour celles qui doivent maintenir plusieurs sites web. Toutefois, l'implémentation des technologies du web sémantique dans un CMS en font un outil de choix pour la gestion de corpus de données

de recherche. Les laboratoires de recherche se tournent vers cette solution pour organiser et rendre visible leurs données.

Nous retrouvons par exemple :

- le Répertoire des écritures numériques initié par la Chaire de recherche du Canada sur les écritures numériques ;
- l'exposition numérique de la librairie J. Willard Marriott de l'Université de l'UTAH. Ce site regroupe plusieurs collections dans Omeka S ;
- les Collections culturelles de l'Université de Tasmanie ;
- La plate-forme numérique EMAN ;
- bien d'autres exemples sont disponibles dans la documentation d'Omeka S à l'adresse suivante : https://omeka.org/s/directory/.

CONTRAINTES

L'installation et la prise en main d'Omeka S est relativement simple. La documentation en ligne est très riche et est largement suffisante pour déployer et utiliser cette technologie sans avoir des compétences particulières en développement informatique.

Omeka S est un outil en développement depuis seulement quelques années. Il présente ainsi quelques faiblesses mineures telles que le nombre limité de *templates* d'affichage pour les sites en ligne, qui sont en nombre très restreints, ou encore des possibilités de manipulations des données indexées à l'intérieur du *back-office* également très limitées. Les modules de visualisation de données intégrés au logiciel permettent de créer uniquement des compteurs mais pas des visualisations plus complexes, or nous aurions aimé pouvoir manipuler les données de manière plus poussée afin d'analyser plus finement les contenus. Toutefois l'API REST à disposition dans le CMS permet de récupérer l'ensemble des jeux de données au format JSON-LD et de les coupler avec d'autres outils ou scripts pour des manipulations plus avancées. Nous pouvons par exemple réintégrer les jeux de données dans D3JS, un framework *open source* basé sur le langage de programmation JavaScript.

PELICAN

DESCRIPTION LONGUE

Pelican est un anagramme pour du « calepin », ce bloc de feuilles dans lequel on écrit. Avec un nom pareil, tout est dit. On sait que Pelican ne servira qu'à produire du texte : l'environnement ne propose pas de fioriture et de fonctions multi-usages pour produire d'autres contenus comme des catalogues de fiches produits (e-commerce-). C'est un générateur de sites statiques remontant à la première génération. Sa première version mise en ligne date de 2010. Ce générateur est toujours maintenu depuis lors et continu de se doter de nouvelles fonctionnalités (la version de juillet 2022 est la 4.8). Initialement, Pelican a vocation de parser des documents formatés en Markdown ou reStructuredText pour les transformner en fichiers HTML.

Pelican s'installe avec l'installateur de paquet pip et utilisable via une série de lignes de commandes.

L'écosystème basique de Pelican peut être amélioré via une série de plugins mis à disposition sur le répertoire GitHub du logiciel.

LANGAGE DE PROGRAMMATION

Pelican est un générateur de sites statiques très léger (*cf* le code source) qui fait partie des quelques générateurs de sites statiques écrit avec le langage de programmation Python comme c'est le cas pour Nikola ou Sphinx.

INTERFAÇAGE

Pelican a prévu de déployer le site produit sur Netlify ou sur GitHub. Sur GitHub, il est possible d'utiliser l'API pour employer Tina ou Forestry comme CMS pour gérer les contenus.

EXEMPLES D’UTILISATION

Voici deux exemples de sites Web réalisé avec Pelican :

- Le site Web The Digital Cat Books ;
- Le blog de J-O Eriksson.

CONTRAINTES

Pelican répond aux attentes que l’on peut espérer : il propose de gérer du texte et il le fait très bien. La proposition de Pelican est, en dehors de la liste de plugins, minimaliste. Néanmoins, cette structure initiale peut être complétée par tout un ensemble de fonctionnalités via des plugins : des *shortcodes*, un moteur de recherche intégrée, un module multilingue, un module de lecture Asciidoc pour remplacer le format des documents sources, etc.

Toutefois, plus qu’une contrainte, il s’agit plutôt d’une limite, Pelican n’est pas prévu pour produire de multiples formats ou pour gérer des contenus à forte valeur sémantique. Cette limite ne peut être outrepassée aisément car Pelican ne prévoit pas ce cas d’usage (sauf si on récupère les sources en Markdown pour les traiter autrement ; dès lors, il s’agit de monter une forge parallèle à Pelican). En somme Pelican n’est pas prévu pour être implémenté dans une chaîne plus complexe.

QUARTO

DESCRIPTION LONGUE

> The overarching goal of Quarto is to make the process of creating and collaborating on scientific and technical documents dramatically better.

Quarto est un jeune outil dédié à produire des documents scientifiques, qu'ils soient dans des formats numériques ou imprimés (voir l'annonce du lancement en 2022 sur le blog de Posit). Quarto est développé par Posit, le logiciel est maintenu sous licence GNU GPL v2. Quarto se présente comme une surcouche au logiciel de conversion Pandoc. Installé comme un plugin dans son éditeur de texte préféré, Quarto permet de faire toutes les transformations que propose Pandoc et bien plus encore. Ce logiciel intègre la prossibilité de faire créer des présentations (Beamer ou Revealjs), des sites web (des blogs ou des sites vitrines), des livres ou des documents destinés à l'impression. L'écriture des sources reposent sur deux formats principaux : le YAML pour les métadonnées et le Markdown pour le texte, dans sa saveur Quarto Markdown (.qmd). La totalité des projets, s'appuie sur une configuration des paramètres en YAML. Les métadonnées des projets sont souvent dans des fichiers .yaml dédiés, avec les noms des auteurs, les titres, les tables des matières, etc. Le YAML a aussi une autre utilité. Quarto intègre dans les métadonnées des documents les différents paramètres qui vont être utilisés pour la transformtion : on y retrouve le chemin vers le fichier BibTeX contenant les références bibiliographiques, ou vers les feuilles de styles, les templates, etc. Parmi les particularités de cette forge, les utilisateurs ont également la possibilité de générer des contenus dynamiques en utilisant Python, Julia, R, et Observable (Jupyter noterbooks).

LANGAGE DE PROGRAMMATION

Quarto est basé sur la technologie de conversion Pandoc, à laquelle est ajouté un ensemble de scripts écrits avec le langage de programmation TypeScript. Il est possible de contribuer à Quarto depuis le répertoire GitHub du projet.

INTERFAÇAGE

Quarto n’est pas interfaçable comme les autres forges. Pour ce logiciel, accessible en ligne de commande avec Quarto CLI, il s’agit de l’intégrer à un environnement d’écriture pour la rédaction de fichiers sources. Exceptionnellement, cette partie de la fiche de présentation du logiciel sera dédiée à l’installation de ce dernier. Quarto peut être installé en tant que plugin dans différents environnements : R Studio, VS Code (et VS Codium), Neovim et d’autres. Ensuite les documents sont produits en local, sur l’ordinateur de l’utilisateur. Pour les sites web, les développeurs ont prévu plusieurs possibilités de déploiement comme sur GitHub avec le service GitHub Pages, ou sur Netlify.

EXEMPLES D’UTILISATION

Nous pouvons trouver toute une série d’exemples dans la galerie du site web de Quarto (lui-même produit avec Quarto).

CONTRAINTES

Facile à installer et à utiliser, Quarto s’adresse à tous types d’utilisateurs, qu’ils soient débutants ou experts. Le logiciel est très facile à mettre en oeuvre et en quelques lignes de commandes ou clics dans son éditeur de texte, des documents peuvent être produits. Pour des usages plus avancés de Quarto il sera nécessaire de comprendre plus en profondeur les mécanismes du logiciel, ou ceux d’autres outils ou langages, par exemple pour personnaliser la mise en page d’un fichier PDF il est nécessaire de créer un *template* LaTeX. Les résultats sont très satisfaisants pour tous les types de projets.

Nous nous confrontons cependant à plusieurs contraintes. Tout d’abord celles qui sont liées à Pandoc. Pandoc est un outil tout à fait hors norme, toutefois, en faisant de cette technologie le coeur de Quarto, même si Quarto en augmente et en étend les fonctionnalités, il n’est pas possible de dépasser les limites de Pandoc (voir le glossaire).

La contrainte suivante est liée aux langages de balisage léger. L’utilisation de Markdown et YAML limite la richesse sémantique que l’on peut introduire dans un document source et cela malgré l’implémentation de *shortcodes*. Toutefois, en s’appuyant sur les

filtres Pandoc (en LUA ou Python) nous pouvons espérer contourner cette contrainte (et produire de riches documents XML par exemple).

QUIRE

DESCRIPTION LONGUE

Quire est un ensemble de logiciels et de programmes qui forment une chaîne de publication. Cette chaîne est développée et maintenue par le département édition numérique de la fondation/musée J. Paul Getty depuis 2017 pour les premiers prototypes. Quire est pensée pour produire plusieurs formats de sortie à partir d'une même source, tout en privilégiant la pérennité des données (en entrée et produites), la richesse des contenus, la lisibilité des formes produites et l'indépendance quant à son utilisation.

> Quire is a modern, multiformat publishing tool designed for longevity, discoverability, and scholarship. Using a single set of plain text files, Quire creates books as authoritative and enduring as print and as vibrant and feature-rich as the web—all without paying a fee or maintaining a complicated server.
> Quire, https://quire.getty.edu/

Quire consiste en deux éléments principaux : quire-cli et quire-theme. quire-cli est une CLI (*Command-line interface* pour interface en ligne de commande en français) développée en JavaScript qui comporte une série de scripts. Ces scripts génèrent les différents formats de sortie. quire-theme est un thème (ou modèle ou *template*) Hugo (un passage vers le générateur de site statique 11ty est prévu) qui est la structuration et la mise en forme des différents éléments d'un livre. L'interface en ligne de commande, quire-cli, s'utilise, comme son nom l'indique, dans un terminal, afin de lancer les différentes actions comme générer les formats de sortie HTML, PDF et EPUB. Le thème Hugo, quire-theme, est un ensemble de gabarits pour les versions web et PDF, il s'agit donc de fichiers HTML et CSS. Concrètement quire-cli lance plusieurs scripts, qui eux-mêmes déclenchent des programmes pour générer les versions selon plusieurs paramètres. Le générateur de site statique Hugo est au centre, c'est la brique chargée de générer les fichiers HTML et CSS à partir des données en Markdown (textes) et YAML (métadonnées). Ce sont ces fichiers HTML/CSS qui vont permettre ensuite la génération des formats PDF et EPUB. Pour le format PDF Quire utilise Prince XML, un processeur qui transforme une série de fichiers HTML en un fichier PDF paginés, cela à partir de feuilles de style CSS.

Quire est une solution clé en main, c'est une chaîne de publication qui offre plusieurs avantages :

- il s'agit de l'assemblage d'un certain nombre de programmes ou logiciels qui ont déjà fait leur preuve ;
- il est possible de modifier un élément de la chaîne sans mettre en péril le fonctionnement global. Par exemple l'équipe qui travaille actuellement sur Quire prépare le passage à un autre générateur de site statique pour remplacer Hugo : 11ty ;
- il est envisageable de reproduire une chaîne similaire mais avec des outils très différents.

Quire vient avec un modèle ou *template* pour l'organisation et la mise en forme des contenus. Ce modèle peut sembler relativement rigide, et répond aux besoins d'un catalogue, mais la structure peut être facilement modifiée pour répondre à d'autres exigences.

Quire dispose d'une documentation détaillée, ainsi que d'une communauté, qui permet à quiconque souhaitant adopter cette solution de trouver de l'aide sous différentes formes.

LANGAGE DE PROGRAMMATION

Quire est principalement basé sur le générateur de site statique Hugo (bientôt remplacé par 11ty), et sur une série de scripts écrits en JavaScript pour automatiser certaines tâches. En parcourant le dépôt Git, il semble que des tests soient en cours pour un possible remplacement de Hugo par 11ty : l'apport pourrait être important puisque l'application dans son ensemble reposerait sur un environnement JavaScript.

À noter que Quire ne nécessite pas un niveau technique avancé pour être utilisé, c'est une chaîne de publication accessible (même si la ligne de commande est requise).

INTERFAÇAGE

Il est théoriquement possible de *brancher* un CMS sur Quire afin d'éditer les contenus (aux formats Markdown et YAML). Attention toutefois, il s'agit bien ici d'*éditer* les

contenus et non d'avoir une prévisualisation ou en rendu final depuis cette interface. Automatiser la génération des formats de sortie (c'est-à-dire déclencher les actions nécessaires à la production des différents formats de sortie, et donc appeler les programmes chargés de ces opérations) via un serveur semble complexe en raison des différents composants, mais néanmoins possible. Ce qui est notamment pour le cas de PrinceXML, un processeur PDF relativement lourd, qui demande une licence (propriétaire) spécifique ainsi qu'un environnement informatique particulier.

EXEMPLES D'UTILISATION

En plus de certains livres du Getty, il est possible de découvrir de nombreux autres exemples sur le site de Quire dans la section *Showcase*).

La plupart des catalogues ont le même fonctionnement qui peut être décrit ainsi :

- une page d'accueil équivalente à une couverture avec un accès à la table des matières ;
- une organisation (très) linéaire avec des liens/boutons pour aller d'une partie à une autre, les contenus/pages/chapitres sont classés par ordre ;
- des liens internes (hypertexte classique) pour aller d'un texte à une fiche descriptive et inversement ;
- deux types de modèles de pages :
    - des textes longs avec quelques images/figures illustratives, des références bibliographiques et des notes de bas de page ;
    - des fiches d'œuvres dans lesquelles l'image prend une place importante. Ces fiches comportent une entête avec des renseignements techniques, des options d'affichage des images avec un zoom et la prise en charge de IIIF notamment.

CONTRAINTES

Facile à installer et à utiliser, Quire ne présente pas de contraintes *importantes* par rapport à d'autres chaînes de publication complexes. Attention toutefois, le fait d'utiliser un terminal et un environnement Node.js peut être compliqué à gérer pour des personnes n'ayant aucun bagage technique.

La principale contrainte est l'usage d'un langage de balisage léger comme Markdown (accompagné de YAML ou TOML pour la description des données) en format d'entrée. S'il

est possible d’étendre Markdown avec des systèmes de *shortcodes* issus de Hugo ou 11ty, la richesse sémantique est toutefois assez limitée. Les formats de sortie de Quire sont contraints, dans le sens où ils sont prédéfinis (HTML pour le Web, PDF, EPUB) et correspondent à des attentes spécifiques.

Enfin il reste la question des versions de Quire utilisées par plusieurs personnes à la fois : une attention particulière doit être portée ici sur le fait de conserver une même version sur tous les postes des personnes travaillant sur un même projet.

SPHINX

DESCRIPTION LONGUE

Sphinx est un générateur de site statique conçu initialement pour générer la documentation du langage de programmation Python au format web. Cet outil a été crée par Georg Brandl et déposé sous la licence libre Berkeley Software Distribution License (BSD). La spécialisation de cet outil sur l'aspect documentation génère par elle-même les avantages et les inconvénients d'un tel outil. L'inconvénient majeur est qu'il sera presque impossible de créer des blogs ou des sites plus complexes, ou encore d'intégrer et de gérer des collections d'oeuvres numériques. Toutefois, l'outil prend en charge une multitude d'extensions très utiles pour la production scientifique d'une manière similaire à ce que propose le Jupyter NoteBook. D'ailleurs le thème « Book » proposé par Sphinx sert de base pour la plateforme Jupyter Notebook qui permet de produire des livres web.

TYPES D'USAGES

Sphinx est principalement destiné à la création de site de documentation, comme par exemple avec la documentation de Python 3. Les thèmes standards proposent de centraliser les informations sur une seule page avec un menu de navigation sur la gauche de l'écran et les informations textuelles au centre de l'écran. De plus en plus d'utilisateurs font appel à Sphinx pour éditer des livres au format web, d'ailleurs ce format sera réutilisé par les plateformes GitBook et JupyterBook.

TECHNOLOGIES

Les technologies utilisées par l'environnement Sphinx sont Docutils et Jinja.

LANGAGE DE PROGRAMMATION

Sphinx a été développé avec le langage de programmation Python 3 (3.6+).

EXEMPLES D’UTILISATION

Voici quelques exemples de documentations utilisant Sphinx :

- https://docs.python.org/3/
- https://jupyterbook.org/intro.html
- https://inria.github.io/scikit-learn-mooc/
- https://manual.calibre-ebook.com/
- https://inkscape-manuals.readthedocs.io/en/latest/

CONTRAINTES

Sphinx a été pensé pour gérer des collections de textes. C’est un outil simple à installer et à utiliser. La principale limite concernera la gestion des documents autres que du texte, comme par exemple les documents multimédias, qui n’ont pas été pensés comme élément central de cette forge.

STYLO

DESCRIPTION LONGUE

Stylo est un éditeur de texte sémantique développé par la Chaire de recherche du Canada sur les écritures numériques. Stylo est une solution SaaS (*Software as a Service*) hébergée par la TGIR des Humanités numériques Huma-Num, et centralise une multitude de fonctionnalités au coeur des enjeux de l'édition scientifique. Stylo permet, en plus de l'activité initiale d'écriture en Markdown, d'éditer les métadonnées d'un document, de suivre les évolutions d'un document grâce au versionnage de fichiers, de gérer les données bibliographiques (notamment via Zotero ou en BibTeX), d'annoter les documents, de les partager avec des collaborateurs et, enfin, de les exporter dans différents formats. Cette dernière propriété repose sur *Pandoc* pour convertir les trois sources (YAML pour les métadonnées, Markdown pour le texte et BibTeX pour la bibliographie) vers les différents formats de sortie (*output*), ajoutant ainsi Stylo aux forges orientées *Single Source Publishing*.

Une des particularités de Stylo par rapport à d'autres forges est la granularité, Stylo gère des documents de type article ou chapitre. Le module d'export permet en partie de construire des objets éditoriaux plus complexes comme des livres.

LANGAGE DE PROGRAMMATION

Stylo repose sur différentes technologies :

- ReactJS avec Node.js pour le *frontend* ;
- MongoDB pour la base de données ;
- GraphQL en *backend* pour l'API ;
- Python3 pour le module d'export des documents ;
- Pandoc pour les conversions.

INTERFAÇAGE

L'interfaçage de Stylo avec d'autres outils n'a pas encore été développé. Toutefois il serait intéressant d'utiliser Stylo comme premier chaînon d'une forge pour éditer des

contenus suivant le principe du *single source publishing*.

EXEMPLES D'UTILISATION

La revue Sens Public a intégré Stylo dans sa chaîne éditoriale. La revue Romanticism on the Net a également intégré Stylo dans sa chaîne éditoriale.

CONTRAINTES

La prise en main de Stylo ne pose aucune contrainte spécifique : il n'y a rien à installer en local puisque l'outil est mis à disposition en SaaS sur le Web. Pour utiliser Stylo, chaque utilisateur doit se créer un compte, soit via Stylo soit via les services de la structure Huma-Num, afin de profiter pleinement des fonctionnalités de l'application et du stockage des documents sur les serveurs de Stylo.

Stylo devient un environnement d'édition intégré dans une page web. Toutefois, en dehors du système de tags, l'outil ne permet pas encore d'organiser des documents qui vont rester à plat dans une liste. Comme il est mentionné dans la présentation de l'outil, Stylo est un éditeur de texte et n'a donc pas vocation à prendre en charge d'autres documents orientés multimédias.

La communauté qui gravite autour de Stylo est d'une taille modeste si on le compare à d'autres éditeurs de texte (ou outils de traitement de texte) comme Microsoft Word. L'intérêt majeur de Stylo réside dans la proposition d'une autre vision du monde, différente de celle de Microsoft Word qui domine le marché depuis bientôt trois décennies. Cette vision du monde remet en question des pratiques intégrés par habitudes et peut devenir un frein puisqu'elle impose une forme d'apprentissage à travers une déconstruction des acquis suivi de l'acceptation d'une nouvelle vision du monde : une matière donnée à penser et à digérer pour la faire sienne.

Enfin, Stylo est une porte d'entrée vers d'autres approches possibles de l'écriture numérique : un usage avancé conduira certaines personnes à utiliser un éditeur de texte sur leur ordinateur et Pandoc en ligne de commandes, ou à utiliser un logiciel comme Zettlr.

REGARD CRITIQUE

Après avoir présenté une série de chaînes de publication et les contraintes liées à ces outils d'édition et de gestion de contenu, nous devons désormais porter un regard critique sur les points que nous considérons comme saillants. Comme mentionné dans l'introduction, ce rapport n'est pas un travail exhaustif mais donne à voir une situation actuelle liée à un problème spécifique : comment gérer, convertir, transformer, organiser des contenus textuels et multimédias dans une démarche éditoriale et muséologique.

DES FORGES ET DES CONTRAINTES

Les forges présentées révèlent à la fois une richesse, une diversité et une disparité de fonctionnalités liées à des objectifs parfois très différents. Ce que nous avons présenté, c'est le fait que les forges sont le produit de leurs usages, c'est-à-dire que les outils tels qu'ils sont pensés sont souvent détournés pour des usages très spécifiques. Le rapport s'est beaucoup intéressé à ce que nous appelons des générateurs de site statique, outils à l'origine pensés pour gérer des blogs ou des sites web peu complexes. Pourtant, les exemples que nous avons donnés – **cf.** les fiches des forges – démontrent la relative plasticité de ces solutions. Les limites ont été pointées dans les contraintes et concernent principalement les questions de compatibilité, de format et d'obsolescence. La dépendance est probablement le point d'articulation entre des systèmes autonomes et des forges qui peuvent bénéficier d'un écosystème en constante évolution. Du point de vue des formats, la question du balisage est centrale. Nous avons volontairement laissé ouvert les possibilités au-delà du langage XML.

Les initiatives éditoriales que nous avons analysées ont toutes comme point commun de rassembler un certain nombre de programmes, d'outils et d'expériences afin de gagner du temps et de bénéficier des initiatives passées. Ce travail d'assemblage, de configuration et en partie d'écriture de code est donc dicté par un certain nombre de contraintes, contraintes qui sont parfois inhérentes aux programmes, aux outils ou aux valeurs portées par l'équipe de gestion du projet. La particularité des projets éditoriaux évoqués ou analysés est le positionnement des porteurs de projet : il ne s'agit pas d'entreprises qui visent à créer *ex nihilo* des outils commercialisables en fonction de besoins mais d'équipes pluridisciplinaires aux forces et aux budgets limités. En d'autres termes, l'enjeu est de pouvoir réutiliser des programmes déjà existants, soit parce que ceux-ci répondent déjà aux besoins, soit parce que le temps nécessaire à créer *from scratch* ces outils n'est pas disponible.

Les contraintes d'usages, techniques et sociales ainsi identifiées permettent de prendre en compte les obstacles liés aux outils existant, mais aussi de cadrer les objectifs dans l'écosystème numérique influencé par le Web. En conséquences, nous prenons le parti de considérer les contraintes comme genèse d'un projet éditorial : ce sont elles qui, une fois explicitées, permettent de définir les besoins et les choix à effectuer dans un projet.

AU CŒUR DE LA FORGE

Nous pouvons remarquer qu'il y a au minimum deux motifs de contraintes différents, un motif micro et un motif macro. La contrainte peut intervenir à plusieurs échelles d'un projet, par exemple à celle d'une brique de la forge, ou plus largement sur plusieurs maillons voire sur l'entièreté de la forge. L'influence des contraintes peut également dépasser la forge elle-même, être à l'échelle de l'écosystème du Web et des technologies qui y sont liées. Nous nous intéressons désormais au plus petit motif, le micro, celui interne à la forge.

L'une des premières questions qui se pose lors de l'établissement d'une forge est celle des objets à créer ou à produire. Chacun de ces objets est livré dans un format spécifique, et chaque format implique une technologie qui lui est propre. De fait, si nous voulons axer la chaîne éditoriale sur le modèle/principe du *Single Source Publishing*, il faut penser à agencer les technologies — les briques — entre elles. L'enjeu qui vient donc d'être soulevé ici est celui de la compatibilité. La première limite que nous pouvons constater est celle de la constitution d'une forge dont les briques sont compatibles entre elles. Pour rappel, la compatibilité n'est pas juste une affaire de traitement de plusieurs formats, mais la capacité à préserver la richesse sémantique d'un document et ce, quelles que soient les opérations qui lui sont appliquées. Cette limite se traduit par des choix qui peuvent nous ramener à des chaînes linéaires, c'est-à-dire qu'un manque de connaissances techniques, de temps, de budget (ou tout autre influence) peut orienter le choix vers la facilité, soit vers des chaînes linéaires et préexistantes et donc qui ne sont pas adaptables et généralement non modulaires. Cela nous amène à la question de la littératie numérique, enjeu qui est finalement primordial dans ce rapport : la plasticité et la modularité, soit la capacité à changer des briques selon les besoins, nécessitent un apprentissage technique et une montée en compétences des gestionnaires du projet.

Une autre limite que nous pouvons soulever est celle de la liaison des contraintes entre elles. L'établissement d'une chaîne multimodale ne soulève pas que des enjeux techniques, comme nous avons pu le mentionner dans la typologie des contraintes, mais également des contraintes sociales et d'usages, or celles-ci ne sont pas décorrélées les unes des autres. Par exemple, comme nous venons de le voir, la contrainte des formats est intimement reliée à la contrainte de compatibilité. D'une certaine manière, les contraintes qui se trouvent à l'échelle la plus grande sont héritées de l'échelle la plus petite, et si nous filons cet exemple, notre échelle de contrainte la plus petite semble être la communication des informations entre deux objets donc entre deux formats, contrainte qui sera nommée compatibilité à l'échelle de la forge, et interopérabilité à l'échelle technologique.

LA ZONE AU DEHORS

Aujourd'hui, les enjeux technologiques autour du Web concernent l'interopérabilité des données et des informations qui circulent sur ces réseaux. Il s'agit donc de faire en sorte que les informations produites avec une forge puissent être réutilisées, lues, interprétées et réconciliées avec d'autres informations extérieures à la forge. L'une des réponses proposées par plusieurs communautés se trouve être l'utilisation de standards ouverts et libres. Cependant, les formats préconisés pour cet enjeu de l'interopérabilité s'apparentent aux formats utilisés pour nos sources primaires et non à ceux des artefacts produits. La limite entre l'intérieur de la forge et le monde technologique pose la problématique suivante : dans quelle mesure est-il possible de rendre interopérable les artefacts produits par des forges ?

Depuis le début de ce rapport, nous sommes revenus à plusieurs reprises sur la nécessité de préserver la richesse sémantique d'un document lors des transformations qui lui sont appliquées. Si nous avons autant insisté sur cette contrainte, c'est parce qu'elle n'apparaît pas ou peu lors de la création d'une forge. Or, si des informations sont vouées à être interopérables, il faut le penser en début de projet, pour ne pas avoir à retravailler les objets produits une fois qu'ils sont finalisés. Une deuxième limite, finalement peu abordée à l'occasion de ce rapport, est celle de la valorisation. Schématiquement, la valorisation se distingue en deux axes : l'un concerne le référencement, c'est-à-dire la mise à disposition de données structurées pour des machines ; et l'autre la communication mise à disposition au lectorat humain. Le premier axe relève d'un assemblage technique ou d'un moyen pécunier qui peut être solutionné en ciblant les plateformes sur lesquelles nos objets seront diffusés. Dans les forges que nous

avons présentées, nous ne nous sommes pas arrêtés sur des modules spécifiques/particuliers liés à ce référencement, toutefois il est possible de répondre à ce besoin avec les outils mentionnés. Pour le deuxième axe, la communication autour des différents objets forgés est bien souvent trop limitée, conséquence du manque de temps et de budget mentionné précédemment. Il nous semble qu'à cet endroit se trouve un enjeu décisif pour ouvrir le plus largement possible les productions.

LE CAS DU XML

Pour ce rapport, la demande porte également sur la constitution d'une chaîne pouvant prendre en charge un format d'entrée comme le XML, et plus spécifiquement des schémas TEI. Nos recherches n'ont pas abouti concernant des forges constituées autour de XML, si ce n'est des logiciels créés sur mesure et fondés sur des transformations complexes basées sur XSLT ou BaseX. Des modules prenant en compte le format XML et s'inscrivant dans l'écosystème du développement web existent pourtant, tel que SaxonJS — exécutables en *front* ou en *back* avec Node.js. Il serait donc possible d'imaginer une forge fonctionnant avec Node.js, avec un préprocesseur comme SaxonJS assurant la conversion vers HTML (pour des sorties web ou paginées via Paged.js) et des générateurs de site statique comme 11ty ou SvelteKit pour la production des artefacts.

Il est étonnant qu'aucune initiative dans ce sens ne semble émerger, tant les composants semblent désormais prêts à être réunis. Nous proposons plusieurs hypothèses. La première hypothèse est celle de la rencontre de deux univers et communautés qui ne semble pas se faire : les projets qui ont comme source des documents XML avec des schémas parfois complexes sont assez éloignés des logiques du web (modularité, usage de modèles/*frameworks* récents, documentation du code), n'éprouvant pas la nécessité de *montrer* les rouages inhérents. La seconde tient dans le fait que l'édition de fichiers XML est fortement liée aux quelques logiciels disponibles : oXygen est l'un de ceux qui est le plus utilisé (à raison), limitant l'interconnexion (voir l'usage) avec d'autres interfaces graphiques comme des applications web facilement configurables.

XML reste encore largement utilisé dans des chaînes d'édition créées sur mesure pour réaliser un travail de conversion d'un balisage à l'autre. Les chaînes d'édition XML sont très puissantes pour générer des formats de sortie, mais elles sont rarement pensées pour prendre en charge la dimension *publication* (produire un site web organisé, envisager la gestion de corpus, etc.). D'une certaine façon, XML subit un éloignement de communautés qui s'orientent vers des solutions plus souples, préférant l'ouverture

des *frameworks* contemporains (pour le meilleur et pour le pire) à la rigidité (toute relative) de XML.

CONCLUSION

L’objet de ce rapport est de porter un regard sur la constitution de chaînes de publication, ici appelées forges, qui permettent de produire des artefacts de types livres, sites web et documents présentant des collections d’œuvres ou d’objets artistiques. La problématique qui nous a animé tout au long de l’écriture de ce papier est relative à la pérennisation des collections de documents à travers les processus d’édition des divers artefacts mentionnés. Parmi toutes les opérations de traitement des informations, l’enrichissement sémantique fait partie intégrante de ces chaînes de publication. Pour ce faire, les forges doivent prendre en compte des balisages complexes et produire des objets numériques qui conservent cette richesse sémantique. Ces forges doivent également permettre une manipulation et une valorisation des contenus produits.

Nous avons construit le rapport autour de forges sélectionnées selon des critères tels que leur nature (éditoriale), le caractère libre ou *open source*, le recours à des standards et la modularité. Chaque forge a fait l’objet d’une analyse selon une grille précise nous permettant d’établir une liste de contraintes. Ce travail nous a permis de dresser un état de l’art des solutions d’édition numérique ouvertes et en grande majorité interopérables. Loin de pouvoir définir la solution répondant aux besoins de CIÉCO, nous avons pu frayer plusieurs pistes de réflexion permettant de constituer un panorama utile.

Le rapport nous révèle un angle mort dans l’écosystème exploré : l’enrichissement et le traitement des sources primaires serait pertinent avec un langage de balisage lourd (XML) car celui-ci est beaucoup plus verbeux et précis pour la description d’éléments, seulement nous trouvons en majorité une prise en charge des langages de balisage léger au détriment d’autres solutions qui sont encore rares et peu modulaires. Il est difficile de pouvoir combiner plusieurs des éléments suivants : une gestion aisée des contenus (écriture, édition, organisation), une prise en compte des multimédias (typiquement IIIF), une richesse sémantique accompagnée d’une logique de schéma (XML/TEI) ou la constitution d’applications web permettant la curation de contenus dans les espaces numériques. Si nous considérons que ce rapport à un angle mort principal, c’est clairement la prise en compte de forges qui permettent la gestion de données encodées en XML. Une telle solution est encore à inventer, le problème étant qu’il semble impossible de bénéficier de la souplesse et de la puissance des technologies du web dans cette perspective.

En creux de ce rapport émerge deux thématiques : *low-tech* (Mateus & Roussilhe, 2023) et *minimal computing* (Risam & Gil, 2022). L’écueil qu’il nous semble nécessaire d’éviter dans la constitution de forges est celui de la suringénierie : l’usage de technologies

peu accessibles (qui requièrent des connaissances techniques avancées) à travers un agencement complexe (difficile à appréhender et à maintenir). La réalisation des objectifs (la production d'artefacts) n'est pas le seul critère dans le choix et la sélection de forges, les *façons de faire* sont également déterminantes, politiques et modèlent nos façons de pensée. La facilité de compréhension d'une forge et de ses briques ainsi que la possibilité de l'adapter sont pour nous un prérequis indispensable.

L'écriture de ce rapport a été l'occasion d'explorer des technologies qui sortent de notre champ habituel et de les confronter à nos pratiques pour comprendre comment les intégrer et les agencer à l'intérieur d'une forge. Nous nous sommes parfois heurtés à des difficultés d'implémentation, et il nous semblait complexe d'utiliser ces outils sans devoir les tordre et les déplacer de leur application initiale. Nous n'avons pas retenu ces agencements trop complexes qui sortent du cadre de ce rapport puisqu'ils nécessitent des compétences en développement bien plus avancées et font preuve d'une quasi-absence de modularité ou d'interconnexion avec d'autres briques existantes. Une dépendance trop forte à un outils ou une brique trop complexe à l'intérieur d'une forge génère le risque d'une régression de la chaîne modulaire vers une chaîne linéaire et monolithique. Par trop complexe, dans ce cas-ci, il faut y entendre une brique qui n'est pas maîtrisée. Si une forge formée avec une brique complexe est dépendante de cette dernière, la forge perd d'abord en modularité (ou alors il faut trouver une autre solution à cette brique) puis s'expose à se transformer en chaîne linéaire : des contraintes d'usages apparaissent et peuvent limiter les actions possibles jusqu'au point où cette brique n'est utilisée que pour une seule opération à un seul moment du flux de la forge. Quelles que soient les connaissances requises dans l'élaboration et le déploiement d'une forge, il devient nécessaire de comprendre et d'apprendre les fonctionnements techniques et leurs enjeux afin d'augmenter le pouvoir d'agir des concepteurs et usagers de la forge (Vallerie & Le Bossé, 2006).

CAS D'ÉTUDE

Nous voici à la fin de ce rapport. Nous souhaitons ajouter quelques lignes supplémentaires pour aborder la phase de conception d'un projet d'édition souvent sous-estimée, alors que de cette phase découle l'ensemble des agencements des briques de la forge de publication. Cette dernière partie traitera d'un cas fictif de mise en application du rapport qui précède : les différents éléments présentés sont ici testés avec ce projet imaginé, afin d'épuiser les principes, de confronter les contraintes à un projet complet et de dessiner quelques perspectives en lien avec les différentes parties du rapport.

De notre point de vue, la conception d'un projet d'édition ne repose pas juste sur l'idée d'un ou plusieurs objets à publier mais sur les choix de l'infrastructure technique à concevoir pour atteindre les objectifs souhaités. Finalement, cette phase est un moment charnière du projet : la théorie de l'éditorialisation telle qu'elle est mentionnée au début de ce rapport nous rappelle que ce sont les choix techniques et méthodologiques qui permettent à l'objet produit de faire sens et d'être l'incarnation d'une pensée singulière (Vitali-Rosati, 2021).

CADRE GÉNÉRAL DU PROJET FICTIF

Le cadre général sera le suivant : une Encyclopédie de la pomme, dont la source est un ouvrage imprimé au 18^{e} siècle, cette encyclopédie doit être mise en ligne sous la forme d'un site web, mais aussi mise à disposition via une application web et une version imprimée (non fac-similé). L'enjeu est de permettre la navigation dans plusieurs centaines de fiches descriptives de variétés de pommes et d'éléments historiques, et aussi d'ajouter des textes permettant d'introduire des parties théoriques et des contributions supplémentaires. Les différents artefacts qui seront produits doivent montrer toute la richesse de l'encyclopédie tout en sélectionnant les parties les plus pertinentes, et en trouvant des moyens de rebond entre les contenus.

L'équipe porteuse du projet est composée de trois chercheur·e·s spécialisé·e·s chacun·e en agronomie, en humanités numériques et en édition numérique. L'objectif principal du projet de recherche est de publier une édition numérique de l'Encyclopédie de la pomme dans trois formats/artefacts différents : un ouvrage imprimé, une édition web augmentée sous forme de site *classique* et sous la forme d'une *web application*. La dénomination « édition web augmentée » désigne un format qui tire profit des navigations de lecture multidirectionnelles réalisables dans le texte grâce à l'hypertexte et à HTML, chose plus difficilement permise par le papier. Les contenus peuvent ainsi être augmentés

(liens hypertextes, injection de contenu, création d'index et de glossaires), liés à une taxonomie et mis à disposition pour un moissonnage. En terme d'interface, l'objectif est aussi de jouer avec les effets de mise en page et de déclenchement des événements (affichage à l'écran). La gestion des images est importante puisque des gravures et quelques aquarelles font partie de l'ouvrage et sont numérisées en haute qualité.

LES BESOINS ET LES CONTRAINTES

Commençons dès maintenant par lister les besoins et les contraintes pour construire le socle de notre forge de publication. Du côté des besoins nous aurons à piloter trois sorties différentes (sans compter les éventuels jeux de données) pour un même texte : deux versions HTML hébergées sur un serveur web et une version imprimable. Nous pouvons d'ores et déjà nous orienter vers une forge basée sur le principe du *Single Source Publishing* : produire plusieurs artefacts à partir d'une source unique.

Enfin, un projet de traduction de l'encyclopédie viendra compléter la version originale. Cette partie supplémentaire n'est pas traîtée dans l'exemple mais nous ferons mention des paramètres à prendre en compte pour la faisabilité de cette extension. La langue principale de publication sera le français (comme l'encyclopédie) mais nous aurons également des éléments en anglais et en arabe.

Les contraintes peuvent être listées selon la typologie mentionnée dans le rapport : sociales, techniques et d'usage. Toutes les contraintes nommées dans le rapport peuvent concerner l'ensemble des projets d'édition, toutefois, nous ne sélectionnons que celles qui semblent les plus importantes dans le cadre de ce projet.

- les contraintes sociales :
    - libre et *open source* ;
- les contraintes techniques :
    - format ;
    - compatibilité ;
    - multilinguisme ;
    - interopérabilité ;
- les contraintes d'usage :
    - déploiement ;
    - prise en main.

La contrainte d’utilisation de logiciels libres et *open source* émane de l’équipe en charge du projet. La volonté des chercheur·e·s est de promouvoir une science libre, accessible et ouverte ; pour que le projet soit en phase avec les valeurs portées par l’équipe de recherche il convient donc d’utiliser des outils qui vont dans ce sens. L’usage de technologies ouvertes et libres ouvre la possibilité d’une réutilisation par la communauté scientifique et une diffusion des outils dans la communauté plus spécifique des humanités numériques. Les contraintes techniques quant à elles proviennent de la nature des livrables à réaliser. En effet, un livrable sera imprimé, il nécessite de passer par une phase de mise en page, et les deux autres livrables seront mis en ligne sur le Web, donc nécessairement encodés selon les formats du Web (HTML, CSS, JS). La contrainte de format sous-entend deux des trois autres contraintes techniques, à savoir : la gestion de la compatibilité entre les différents formats au sein du projet afin de pouvoir produire nos trois livrables ; l’interopérabilité de nos livrables, au moins pour les versions web, afin que les données puissent être réutilisables par d’autres personnes. La contrainte liée au multilinguisme concerne autant les éléments de navigation de l’édition numérique que l’organisation des versions anglaises et arabes dans un second temps. Le dernier type de contrainte, les contraintes d’usages, concerne majoritairement le niveau de littératie numérique de l’équipe du projet. Nos trois chercheur·e·s vont piloter une équipe composée de différents profils, certains plus orientés vers l’informatique que d’autres, comme par exemple avec une spécialisation en XML. Le dépassement de ce type de contrainte n’est pas uniquement du ressort de l’apprentissage mais l’est aussi de la capacité à former des équipes polyvalentes et pluridisciplinaires.

LES ÉTAPES PRÉALABLES DU PROJET

Avant de déterminer les éléments de conception de la forge, voici les étapes préalables :

1. numérisation de l’ouvrage ;
2. océrisation des pages et découpage des images au moyen d’un logiciel tel qu’eScriptorium ;
3. balisage du texte océrisé avec un schéma XML-TEI ;
4. nommage et organisation des images des illustrations selon le protocole IIIF.

QUI INTERVIENT SUR LE PROJET ?

Trois profils différents sont amenés à intervenir sur ce projet d'édition :

- les personnes en charge du balisage du texte, spécialisées en XML-TEI : profil technique avancé pour le XML ;
- les personnes qui vont organiser les contenus et qui vont proposer des parcours de lecture : profil non technique ;
- les personnes qui pilotent le projet et qui interviennent sur l'architecture globale : profil technique avancé en XML et en technologies du Web.

L'enjeu ici est de faire cohabiter ces trois profils autour d'outils permettant la constitution d'une édition augmentée, avec des accès et des interfaces adaptées : les fichiers XML-TEI sont modifiés puis versionnés sur un dépôt commun ; des interfaces graphiques permettent l'organisation des contenus et la constitution d'expériences de lecture ; des accès aux fichiers de configuration du projet ou de la forge.

CONCEPTION DE LA FORGE

Voici les différentes briques nécessaires pour la réalisation des objectifs :

- un convertisseur XML-TEI vers HTML, tel que SaxonJS ;
- un serveur pour stocker les images numérisées ainsi qu'une API pour les servir selon le protocole IIIF ;
- un générateur de site statique pour convertir les textes additionnels et organiser l'ensemble des contenus (pour les trois formats), tel que 11ty ;
- un module d'impression PDF depuis une page HTML, tel que Paged.JS ;

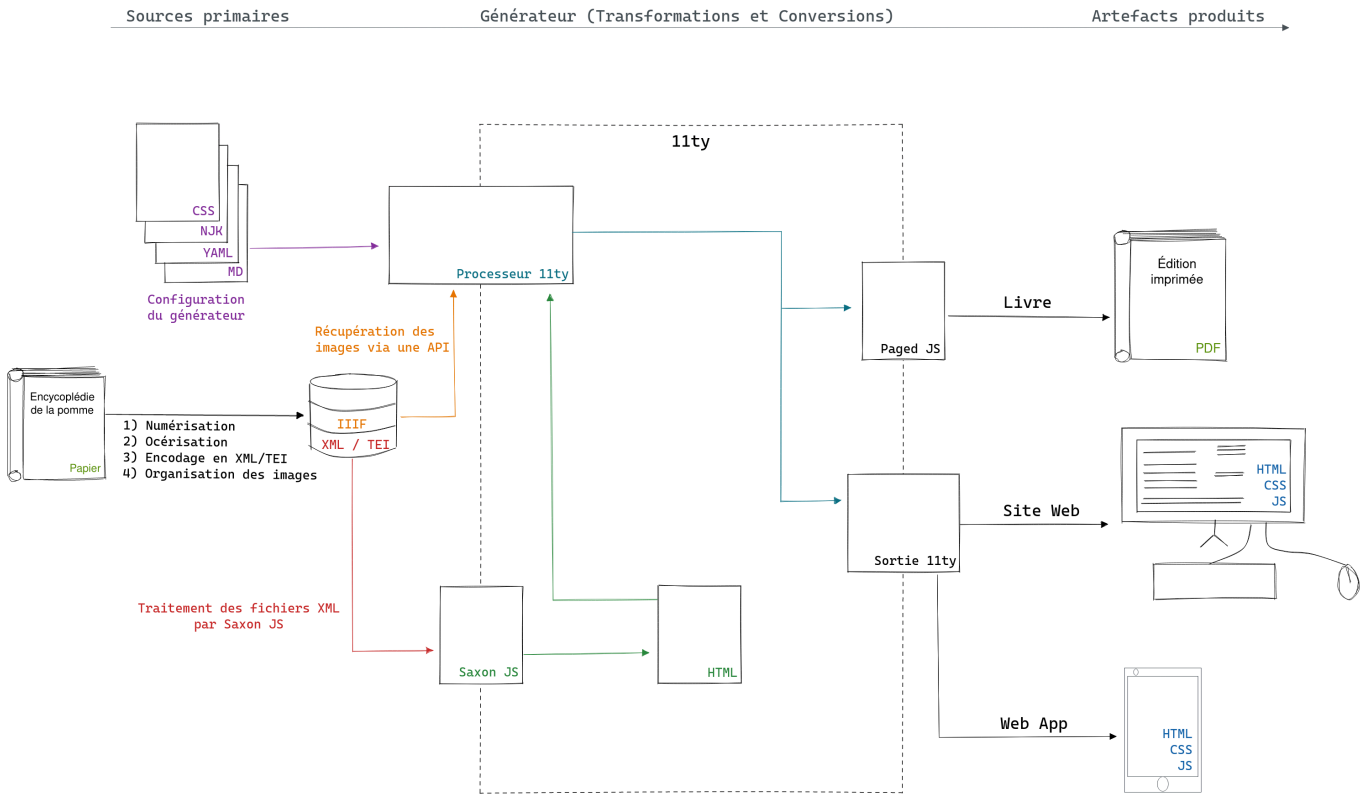

Figure 1. Schéma de la forge

À l'intérieur de cette forge, nous remarquons qu'une des briques, SaxonJS, est propriétaire et non *open source* comme cela est indiqué dans la fiche technique de l'outil. Le problème est qu'à notre connaissance il n'existe pas d'équivalence *open source* en dehors de l'initiative Apache Xalan Project gérée par la *Apache Software Foundation* dont l'implémentation dans une forge de publication s'avère complexe. Pour pouvoir processer du XML, la forge présentée est dépendante de cette technologie propriétaire, créant ainsi une contrainte vis-à-vis de la politique de l'équipe en charge du projet qui souhaitait n'utiliser que des technologies libres et *open source*. Une deuxième contrainte apparait, causée par le monopole sur cette technologie : si l'entreprise propriétaire vient à arrêter la maintenance de l'outil, quelle qu'en soit la raison, l'entièreté de la forge viendrait à s'effondrer étant donné la position du processeur XML au début de la forge. Ainsi, il ne serait plus possible d'effectuer des modifications dans les sources primaires le temps de remédier à cette situation. En somme, cette brique fragilise la modularité de notre exemple.

Une fois les quatre étapes préliminaires réalisées, pouvant être englobées sous l'appellation générique de prétraitement des sources et ce afin qu'elle deviennent comput-

ables, il convient de paramétrer l'ensemble des outils pour processer nos sources primaires et obtenir nos artefacts. Tout d'abord, nous avons choisi pour cet exemple le générateur de site statique Eleventy. Il fait partie des outils de la dernière génération et connait un certain succès auprès de différentes communautés en développement web, jusqu'à des institutions comme Getty qui vient d'effectuer sa transition vers cet outil pour son application Quire. La configuration d'Eleventy se fait par le biais d'un peu de Javascript dans un fichier dédié à la configuration, et le reste depuis les fichiers sources en Markdown, JSON, CSS ou Nunjucks. Le générateur sera utilisé pour processer les fichiers en Markdown et en HTML pour produire les différents documents dans les formats adaptés aux livrables. Toutefois, Eleventy ne peut pas traiter directement le XML. Pour contourner cette limite nous avons fait le choix d'implémenter Saxon JS à la forge afin de transformer les sources primaires du XML vers le HTML. Les documents HTML seront ensuite processés par Eleventy avec les autre documents et produiront en sortie un site web et une application web *responsive*. Enfin, le module Paged.js est intégré à la forge pour produire, à partir des documents traités par le processeur d'Eleventy, les documents PDF pour l'édition imprimée.

LE MOT DE LA FIN

Cet exemple n'est qu'une solution possible parmi d'autres : il a vocation à montrer la création d'une forge modulaire à partir d'une énonciation formelle des besoins et des contraintes d'un projet d'édition numérique. Une forge uniquement basée sur un environnement XML est écartée : l'objectif est de pouvoir bénéficier de la souplesse et de la modularité des technologies issues du Web. Une solution *complète* comme Quire n'est pas privilégiée : nous avons besoin de produire des artefacts avec des contenus et des comportements qui ne sont pas identiques (choix des contenus affichés de différentes façons) ; nous avons besoin de disposer de suffisamment de souplesse pour adapter la forge aux besoins initiaux.

Le choix de la décentralisation des opérations effectuées par la forge vers des modules interconnectés demande certes plus d'attention quant à la maintenance du système mais offre également plus de sécurité. Les bénéfices apportés par un tel système sont les suivants :

- chaque module effectue une opération spécifique et, si l'un d'eux venait à dysfonctionner, il serait interchangeable avec une autre brique ;
- l'association entre module et opération permet, en cas de maintenance, de cibler rapidement le point faible de la forge. Par exemple, si un dysfonctionnement

apparaît lors de la génération des fichiers PDF de la version imprimée, il est fort probable que le problème se situe au niveau du module Paged.js ;

- les sources primaires sont préservées. En dehors du traitement effectué par Saxon JS, les fichiers XML ne sont pas touchés par les différentes actions opérées par la forge jusqu'à l'obtention des artefacts ;
- conformément à l'énoncé de cet exemple, la forge ainsi créée permet de générer trois artefacts différents à partir d'une source unique : l'Encyclopédie de la pomme.

GLOSSAIRE

APPLICATION WEB

Une application web, ou son diminutif *web app*, est plus communément connue comme PWA (*Progressive Web App*). La PWA est développée à partir des technologies du web, tout comme les sites web, et contient un document supplémentaire au format JSON appelé le manifeste. Ce dernier permet de déclarer au navigateur qu'il ne s'agit pas d'un site mais d'une application.

Le développement des PWA répond au besoin des développeurs de toucher un public plus large avec une application mobile. En effet, le développement d'application mobile native requiert d'utiliser le langage de programmation correspondant au système d'exploitation du *smartphone*, iOs ou Android. Afin de toucher ces deux principales catégories de consommateurs il fallait développer la même application dans deux langages différents, or, avec les PWA, le développement ne se fait plus qu'avec les technologies du web et réduit drastiquement les coûts liés au développement d'application mobile.

ARCHIVE

Une archive (numérique) est, d'un point de vue étymologique, la désignation d'un lieu institutionnel dans lequel sont stockés des documents. Ce sont les *archontes*, gardien des archives, qui décident des documents à préserver. En tant que gardien de ce lieu de savoir, les *archontes* détiennent également ce que Derrida nomme le pouvoir herméneutique, c'est-à-dire l'octroi aux seuls *archontes* de la capacité à lire et à interpréter les documents archivés (Derrida, 1995). La notion d'archive est fortement liée à un espace physique ainsi qu'au concept d'autorité. Le numérique chamboule ce paradigme par une démultiplication des instances d'archivage, qui ne sont plus forcément étatiques, et la réduction de l'espace physique de stockage nécessaire pour un document. Le fonctionnement du stockage des données provoque un éclatement de l'archive en tant que lieu et remet en cause les autorités gardiennes de ces archives (Monjour, 2018).

ARTEFACT IMPRIMÉ

Les usages de types *impression* ou *print* doivent nécessairement faire l'objet d'un traitement différent par rapport aux usages à vocation numérique. Les supports papiers requièrent des modélisations différentes que celles appliquées aux écrans. Toutefois, le principe de *single source publishing* permet d'appeler les modèles adéquats pour chaque format de sortie, voir même pour chaque format imprimé souhaité.

ASCIIDOC

AsciiDoc est un langage de balisage léger, sur le même segment que Markdown ou reStructuredText, initialement pensé pour produire de la documentation technique. La ressemblance avec le Markdown est frappante et la transition d'un format à l'autre est aisée. Toutefois, une différence majeure existe entre les deux langages : Markdown se décline en une série de saveurs différentes alors qu'AsciiDoc ne comporte qu'une seule version standard.

BASEX

BaseX est un système de gestion de base de données XML basé sur XQUERY et XPATH. Il permet de manipuler et de gérer de larges collections de données formatées en XML. Cette application dispose également d'une interface graphique permettant de visualiser les interactions avec les données. Ce projet est entièrement *open source* et repose sur une licence ouverte BSD.

BIBTEX

BiBTeX est un format standard permettant de décrire des listes de références bibliographiques. Les différentes données contenues dans un fichier sont indexées sous la forme de dictionnaires (clé : valeur), procurant ainsi la capacité d'adapter les listes de références selon différentes normes bibliographiques.

BLOG

Le **blog** est un type de site web répandu lorsqu'il s'agit de publier à fréquence régulière des contenus généralement succincts autour d'une thématique particulière : qu'il s'agisse d'actualité, de recherche, de récits de voyage, etc.

La forme des blogs actuels se positionne à l'opposé des sites web plus complexes et tend à minimiser le nombre de pages pour des contenus typés *one page*.

CONTRAINTE

La contrainte est souvent associée à des entraves ou à des résistances à un processus. Elle est considérée comme un frein, presque un boulet relié à la chaîne et qu'il faut transporter, contourner ou déplacer pour atteindre un objectif. De notre point de vue la contrainte n'est pas si péjorative. Elle est une caractéristique propre aux matériaux utilisés. De fait, en fonction des circonstances, cette caractéristique s'avère bénéfique ou néfaste au processus développé, dans le cas ou cette dernière est néfaste elle est appelée "contrainte". En tant que caractéristique de la matière, la contrainte permet de définir deux aspects de celle-ci :

- les limites de la matière ;

- le sens dans lequel cette matière peut être travaillée. La contrainte s'apparente alors à un type précis de ligne qu'Ingold nomme *guidelines* (2013). Dès lors, le paradigme de la contrainte est modifié : d'une résistance au travail de la matière elle devient un repère à suivre. Un ensemble de ces *guidelines* forment alors une structure matérielle pré-existante à toutes opérations et devient le socle sur lequel le projet sera bâti. Ainsi, définir les contraintes d'un projet revient à faire les choix de conception qui vont définir sa forme.

DOC.X

Les formats .doc et .docx sont les formats privilégiés pour les documents provenant du logiciel Microsoft Word de la suite de logiciels de bureautique *Microsoft Office*. Le format .doc est le format d'origine du logiciel, c'est un format complètement fermé qui ne peut être traité par aucun autre logiciel ou éditeur de texte. Le format .docx quant à lui est un peu plus ouvert puisqu'il embarque la technologie xml.

DOCUMENTATION

La forme *documentation* d'un site web sert généralement à décrire l'ensemble des manipulations, des ressources et des bonnes pratiques nécessaires au bon fonctionnement d'un outil particulier. Le format du site prend traditionnellement la forme d'une page unique composé d'un menu contenant toutes les rubriques sur la gauche de l'écran et du texte dans la partie centrale de l'écran.

DOCUTILS

Docutils, pour *documentation utilities*, est un système de traitement de texte *open source*, basé sur le langage de programmation Python, permettant de traiter la documentation en texte brut dans des formats utiles, tels que HTML, LaTeX, pages de manuel,

OpenDocument ou XML. Il inclut reStructuredText, le langage de balisage en texte brut facile à lire et à utiliser.

EPUB

ePUB est l'acronyme de « Electronic Publication ». Il s'agit du format standard et ouvert destinés aux publications de livres électroniques. ePUB est développé au début des années 2000 et devient un standard à partir de 2007 lors de la sortie de la deuxième version. 2011 voit l'arrivée de la version 3 qui complète les défauts de la version précédente. Depuis ce format est assez stable et ne connaît pas de mise à jour majeure. ePUB 3 intègre les technologies du web telles que HTML5, la gestion des méta-données, le support du langage de programmation JavaScript. L'ensemble de ces outils permet de prendre en charge les problématiques d'accessibilité notamment pour les personnes malvoyantes.

FORESTRY

Forestry est un gestionnaire de contenus (CMS) compatible avec quelques générateurs de site statique. Il offre la possibilité d'administrer le contenu d'un site web statique depuis une interface graphique. Cette application très légère comprend :

- un éditeur de texte en Markdown ;
- une connexion à Git grâce à l'API (permettant ainsi la mise à jour des contenus directement depuis l'interface graphique).

FORGE

Étymologiquement, la forge désigne la *fabrica*, soit un lieu dans lequel le fer est fondu pour être transformé en métal. La forge permet de produire elle-même ses propres outils sans avoir à passer par un autre corps de métier. Ainsi, il devient possible de

créer dans cet espace un cycle d'adaptation constante aux besoins de réalisation par la mise au point d'outils sur mesures fonction des caractéristiques de l'artefact à produire et du maitre à l'oeuvre. Cette « fabrique » véhicule les opérations de conception, de transformation, de production et de transmission. Plus que les différents minerais ou métaux, ce sont les savoirs qui s'incarnent dans la matière et sont éprouvés par les artisans à travers ces quatre étapes.

Dans le cadre de ce rapport, nous utilisons ce terme pour définir des environnements modulaires de conception, de fabrication et de production pour des publications scientifiques. Ce terme nous permet de placer sur un même plan des solutions complètes de publication, des briques techniques spécifiques comme des convertisseurs de formats, ou encore des processeurs de documents. Le terme forge intègre au moins deux dimensions qui répondent au spectre d'étude de ce rapport : la notion de *fabrication*, qui correspond ici au fait de *fabriquer* des objets éditoriaux ; la notion d'*atelier* qui comprend le fait de travailler à plusieurs dans un espace défini où se fabrique des objets. À travers ces deux dimensions, nous souhaitons garder à l'esprit l'image utopique d'une modularité absolue grâce à laquelle il est possible de développer des espaces ajustés aux livrables à créer. La forge déplace le paradigme de la chaîne linéaire vers un système modulaire en plusieurs dimensions, composé de briques indépendantes et pourtant interopérables, permettant des mouvements – ou des opérations – dans différentes directions. Le terme *forge* implique également la notion d'artisanat, avec cette idée d'une maîtrise technique et précise afin de réaliser un travail de qualité.

FORMATS DE FICHIERS SONORES

Les formats des fichiers sonores sont nombreux et ont tous leur particularité. Il faut faire une première distinction entre des formats d'enregistrement dont l'objectif est de capter l'information avec la meilleure qualité possible, des formats d'archivage pour conserver le fichier sans perte et sur le long terme, et des formats de diffusion qui favorisent le poids des fichiers. Voici quelques formats détaillés pour prendre la mesure de la complexité de ce seul champ qu'est le format de fichiers sonores :

- Midi : *Musical Instrument Digital Interface* est un standard technique désignant un protocole de communication, un format, une interface digitale et des connecteurs électriques permettant de relier entre eux une multitude d'objets dédiés à

l'édition, à l'enregistrement et à la création musicale. Ce format standard du monde de la musique a été créé par Dave Smith en 1981 ;
- MP3 : MP3 est le diminutif d'une norme de compression standard (ISO) dénommée MPEG-1 Audio Layer 3 et désigne 3 couches de codage dont la complexité et la performance sont croissantes. Ce format est l'un des plus répandus depuis le début des années 2000 notamment dans le domaine de la musique ;
- FLAC : Free Lossless Audio Codec est un format *open source* développé en 2001 par la Fondation Xiph.org dédié au fichiers sonores. Il est similaire au format MP3 à la différence qu'il n'occasionne que peu ou pas de perte de qualité lors de l'étape de compression.

FORMATS DES FICHIERS IMAGES

Que ce soit pour la prise de vue, le traitement, l'archivage ou la diffusion dans des espaces numériques ou non numériques, la question du format des images est complexe et évolue depuis les débuts de l'informatique. Les formats sont nombreux, et les façons de permettre un accès diffère selon les objectifs. Voici trois formats communs qui présentent chacun leurs spécificités :

- PNG : le Portable Network Graphics est un format ouvert et standard maintenu par le W3C, il s'agit d'un format d'image matriciel sans perte. Ce format est notamment utile pour l'affichage d'images sur le Web, il permet par exemple de gérer la transparence ;
- JPG ou JPEG : le format JPG pour Joint Photographic Experts Group est un format permettant de *compresser* une image afin de réduire le poids du fichier. Ce format est largement utilisé pour la diffusion sur le Web mais aussi parfois pour la conservation (il est possible de régler le taux de compression) ;
- TIFF : le Tag(ged) Image File Format est un format qui permet plusieurs niveaux de compression, dont certains sont sans perte, ce qui en fait un bon candidat pour la conservation.

Aujourd'hui pour diffuser des images sur le Web, que ce soit pour de l'affichage rapide ou pour permettre un accès à des versions de haute qualité, il existe un ensemble de spécifications techniques sous le nom de IIIF. L'International Image Interoperability Framework (IIIF) définit plusieurs interfaces de programmation (ou API pour *Application Programming Interfaces*) permettant de définir un cadre d'interopérabilité pour la diff-

usion et l’échange d’images sur le Web. IIIF est également une communauté de personnes travaillant pour la mise en place de standards autour des images, souvent dans une dimension patrimoniale et ouverte, prenant en compte les besoins d’accès à des fichiers en haute résolution autant que pour un affichage rapide et accessible.

FORMATS DES FICHIERS VIDÉOS

Les formats vidéos posent une question complexe, qui dépasse celle du poids des fichiers (même si c’est un enjeu de taille pour le stockage ou la diffusion) : celle de la bande passante et donc de la vitesse de chargement du côté de l’utilisateur·trice. Voici les trois solutions plébéscitées pour une diffusion sur le Web :

- stockage et diffusion sur un serveur, avec le problème de la vitesse de chargement ;
- utilisation d’un service tiers, avec des plateformes comme YouTube ou Vimeo, chacune présentant des contraintes différentes ;
- installation d’une application prenant en charge certains des paramètres précédents comme PeerTube.

Nous ne détaillons pas les nombreux formats de vidéo.

GIT

Description générale

Git est un protocole de versionnage de documents. Plusieurs instances de ce protocole comme GitHub, GitLab ou Gitea permettent de versionner des scripts ou fichiers textes à partir d’interfaces graphiques ou plus communément à partir de lignes de commande. Le principe de versionnage repose sur la sauvegarde de toutes les modifications apportées à ces scripts sous la forme de *commit*. Ainsi, n’importe quel individu ayant accès au répertoire en question peut revenir sur les versions antérieures, comparer différentes versions entre elles et sécuriser ses données de travail. Au-delà d’un principe d’organisation des documents en usage personnel, Git a été conçu pour travailler collaborativement sur ces répertoires de données. Chaque collaborateur peut cloner le répertoire

de documents en local et initier des modifications qui seront également suivies par toute l'équipe.

Exemples d'utilisation

La plupart des développements informatiques utilisent ce système de versionnage de documents pour organiser leurs contenus, les différentes fonctionnalités à développer ou encore pour maîtriser les bugs et les différentes mises à jour des scripts. Du côté des logiciels *open source* les dépôts sont généralement ouverts pour permettre à toute la communauté de participer à l'évolution du projet ou pour leur permettre de récupérer le code source.

Git n'est plus seulement un outil de versionnage de scripts informatiques. Il sert également à versionner des fichiers textes comme les fichiers en Markdown ou en HTML. Il sera notamment privilégié pour la gestion des contenus des sites statiques !

Nous trouvons aussi quelques utilisations plus originales et marginales qui détournent l'usage traditionnel de Git comme par exemple la maison d'édition suisse Abrüpt qui utilise Git au coeur de son *workflow* pour éditer ses ouvrages.

Contraintes

Git est un outil très puissant s'il est bien maîtrisé. La gestion des versions antérieures sécurise les données mais n'empêche pas de créer des conflits et la suppression de certaines données en cas d'erreur de manipulation au sein d'un projet collaboratif. La courbe d'apprentissage de Git est assez lente si l'on souhaite une utilisation avancée de cette technologie. Elle nécessite notamment une bonne maîtrise du terminal et des lignes de commandes basiques du système d'exploitation de l'utilisateur.

HTML

HyperText Markup Language, HTML pour les intimes, est un métalangage de balisage pour l'hypertexte. Il est majoritairement utilisé pour structurer le texte contenu sur les pages web. Ce langage sera souvent associés à d'autres langages et technologies tels que CSS (Cascading Style Sheets) pour appliquer des styles différents à chaque niveau

de la hiérarchie d'informations et JavaScript pour encoder les interactions avec les contenus.

La singularité de ce langage repose sur la notion d'hypertexte et désigne le mécanisme qui relie les pages web entre elles. Il permet de créer un lien entre deux pages dont l'adresse (Uniform Resource Locator) de chacune sur le réseau est unique. Ainsi, l'ensemble de ces hyperliens forment une toile : le Web.

Depuis sa création en 1989 par Tim Berners-Lee, ce langage ne cesse d'évoluer et d'être amélioré pour arriver en 2014 à une version 5 (HTML5) toujours d'actualité en 2022. Ce format, à considérer comme un standard du web, est privilégié pour la structuration du texte par la majorité des technologies.

INDESIGN

Le format .indd est celui du logiciel propriétaire Adobe InDesign et appartient à la suite des logiciels Adobe. C'est un logiciel de Publication assistée par ordinateur (PAO) largement utilisé dans le monde de l'édition.

JAMSTACK

Concept marketing initié par Mathias Biilmann en 2015, le directeur exécutif de la firme Netlify. La JAMStack définit une *pile* technologique comprenant :

- JavaScript pour les interactions et l'affichage de données *dynamiques* ;
- API pour structurer et interroger les données — via JavaScript ;
- Markup pour le balisage des contenus, afin de simplifier la rédaction et de permettre de *brancher* des interfaces graphiques d'administration.

Les générateurs de site statique sont la partie émergée de ce mouvement technologique. Même si ce concept a été très utilisé par des sociétés qui se sont enrichies dessus, il y a tout de même une certaine critique de ce concept {à reformuler}.

JSON

Le format JavaScript Object Notation ou JSON est un format largement répandu de représentation structurée d'objets et de données comme le XML peut le faire. Toutefois ce format se distingue de son concurrent par une forme moins verbeuse et de fait génère des fichiers moins gourmands en ressources mais également moins précis et descriptifs au niveau des données.

LATEX

LaTeX fait référence au projet qui porte le même nom : LaTeX. Il s'agit d'une application de gestion typographique des contenus spécialisée dans la production imprimée de documents techniques et scientifiques. Ce projet a été créé en 1980 par Leslie Lamport, et est maintenu depuis 1989 par une équipe de bénévole au sein du projet LaTeX3. LaTeX comprend un langage très verbeux et entraîne une courbe d'apprentissage assez lente. Cependant il est possible d'utiliser cet environnement de différentes manières : soit en éditant un document directement en LaTeX, en incorporant les balises manuellement ; soit en utilisant une feuille de transformation pour utiliser LaTeX comme format intermédiaire entre un format de balisage léger (comme le Markdown) pour produire un document au format PDF.

LIVRE-WEB

Le *livre-web* est l'un des formats appartenant à la famille des livres numériques. À l'instar du format EPUBs, le livre-web repose sur les technologies du Web, à savoir les langages HTML et CSS pour formater le contenu et un langage de programmation comme JavaScript ou Python pour la gestion des interactions. Les différences avec le format EPUB sont la portabilité et le support de lecture. L'EPUB est un format portable/exportable qui ne nécessite pas forcément de connexion internet, contrairement

au format du livre-web qui potentiellement a besoin d'une connexion – si ce n'est pas le cas d'autres procédés doivent être mis en place. L'EPUB requiert par ailleurs une application spécifique pour lire le contenu alors que le livre-web est un site web associé à une URL (*Uniform Resource Locator*) et ne nécessite qu'un navigateur pour en explorer le contenu.

MARKDOWN

Mardown est un langage de balisage léger créé en 2004 par John Gruber. Sa syntaxe, beaucoup plus légère et moins verbeuse que le HTML, permet de structurer et de décrire sémantiquement le texte. Il a été pensé pour pouvoir être converti facilement vers d'autres formats comme HTML, LaTeX ou PDF. Markdown se distingue des autre langages de balisages légers car il est déclinable en différentes variantes (ou saveurs). Chacune d'entre elles ajoute une particularité dans la syntaxe Markdown. Parmi les plus populaires, on retrouve :

- CommonMark ;
- GitHub Flavored Markdown (GFM) ;
- MultiMarkdown ;
- Pandoc ;
- Quarto.

MOTEUR DE TEMPLATE

Les moteurs de *template* sont des outils de structuration des contenus qui simplifient la syntaxe et la gestion de la maintenance. Dans le cadre des générateurs de site statique, le moteur de *template* se situe à l'interface entre le texte brut et le rendu HTML visible sur la page web. C'est lui qui dissocie ces deux aspects dans la chaîne de publication.

- Jinja : Jinja est un moteur de template expressif et extensible pour le langage de programmation Python. Des espaces réservés spéciaux dans le modèle permettent

d’écrire du code similaire à la syntaxe Python. Ensuite, le modèle reçoit des données pour rendre le document final.

- ReactJS : ReactJS est une bibliothèque JavaScript développée en 2013 par Facebook pour créer des interfaces utilisateurs.
- Go : Go est un langage de programmation compilé destiné à faciliter et accélérer la programmation.
- Liquid : Liquid est un moteur de template open source écrit en Ruby par Shopify. Il est initialement destiné aux développements d’applications web.
- VueJS : VueJS est une bibliothèque JavaScript open source utilisée pour construire des interfaces utilisateurs.
- Mako : Mako est une bibliothèque de templates développée en Python.

NETLIFY

Netlify est un environnement privé dédié à la construction de sites et d’applications web selon un unique workflow. Ce service, gratuit ou payant, prend en charge l’infrastructure d’hébergement web des application statiques et l’intégration continue des données.

PDF

PDF est l’abréviation de *Portable Document Format* que nous pouvons traduire par « Format de Document Portable ». Ce format a été créé par la société Adobe en 1992 afin de répondre au besoin de préservation graphique des documents et ce, quelle que soit l’application de lecture du document. Les PDF, initialement complètement figé, deviennent au fil des années plus dynamiques et permettent l’implémentation de plusieurs technologies comme l’hyperliens (PDF cliquable) ou la signature électronique. Aujourd’hui le PDF est un format standard et ouvert, maintenu par l’Organisation Internationale de Normalisation (ISO).

PANDOC

Description générale

Pandoc est un logiciel libre de conversion de documents numériques développé par John MacFarlane en 2006. Ce logiciel réalisé en Haskell permet de convertir des documents dans divers formats tels que Markdown, LaTeX, HTML, ReST, docx en HTML, PDF, docx, et bien d'autres. Les conversions ou transformations opérées par Pandoc reposent sur la structuration des contenus. Sur le site web de Pandoc, une liste des formats pris en charge par le logiciel est disponible directement dans l'index du site. On remarque que tous les formats convertibles sont structurés, qu'il s'agisse d'une structuration par insertion de balises (HTML, XML, Markdown, etc) ou des standards spécifiques pour les références bibliographiques (CSL JSON qui propose une structure verticale des données et un système de description des items par clef : valeur).

Pandoc n'utilise pas d'interface graphique autre que celle du terminal, c'est un logiciel dont les fonctionnalités sont accessibles par lignes de commande.

Exemples d'utilisation

Les usages de Pandoc peuvent être très divers, allant de l'usage personnel à l'implémentation de Pandoc comme brique d'une forge comme c'est le cas pour Quarto ou encore dans Stylo.

Contraintes

Pandoc est un outil très puissant, au point où il sert de fondation à de multiples forges pour transformer et convertir des fichiers sources vers les artefacts souhaités.

La contrainte principale que nous rencontrons avec Pandoc est le langage de programmation utilisé pour le développer :Haskell.

C'est un langage qui propose une autre vision de l'informatique et du langage machine dans laquelle les effets de bords n'existent pas. En conséquence il devient difficile de pouvoir y contribuer si l'on n'est pas familier de cette approche. L'on peut rétorquer à cette contrainte que Pandoc intègre des filtres personnalisables en LUA ou en

Python dans ces commandes, ce qui permet d'en étendre les fonctionnalités et ainsi d'enrichir les documents transformés. Toutefois, ce sont des solutions locales, moins stables et moins pérennes que des fonctionnalités implémentées et maintenues dans un logiciel par une communauté.

PLAIN TEXT

Plain Text est le terme employé pour désigner des données stockées dans un document à travers des caractères visibles et sans représentation graphique ou balisage sémantique des caractères.

PROTOCOLE IIIF

Le protocole IIIF, *International Image Interoperability Framework*, désigne un assortiment de spécifications techniques visant à définir un cadre interopérable destiné à la diffusion de documents multimédias (images et images en trois dimensions) sur le Web. Le principe derrière IIIF repose sur les technologies des API (*Application Programming Interface*). Afin de rendre interopérable les contenus des bibliothèques numériques, IIIF propose différentes APIs pour questionner, afficher ou rechercher des documents dans ces bibliothèques numériques.

RESTRUCTUREDTEXT

Le format reStructuredText (RST) est un langage de balisage léger, concurrent du MarkDown. Au même titre que ce dernier, il est possible de convertir RST dans d'autres formats tels que HTML, XML, LaTeX et ODF. Ce format, moins courant que Markdown, est utilisé comme analyseur syntaxique de Docutils, outils dédié à la documentation de scripts en Python.

SAXON

Description générale

Saxon est un ensemble d'outils pour traiter des documents XML. Il regroupe notamment les principaux processeurs de XML tel que : XSLT, XQuery, XPath et XML Schema. L'intérêt que nous portons à cette technologie dans le cadre de ce rapport concerne surtout le processeur XSLT (*Extensible Stylesheet Language Transformations*). XSLT est un langage basé sur XML et utilisé pour la transformation de documents XML. Ce n'est pas un "convertisseur" mais il permet toutefois de transformer un document formaté selon un langage de balisage (HTML, XML) vers un autre langage de balisage (HTML, XML ou Markdown). Il permet ainsi de créer des pages web à partir d'un fichier en Markdown ou en XML, ou encore d'obtenir un fichier pdf en lui adossant d'autres technologies comme TeX. C'est un langage très puissant permettant de maintenir une chaîne de travail au format XML.

Saxon est adaptable en fonction des besoins et se décline en plusieurs éditions et sur plusieurs langages de programmation :

- SaxonJ est écrit en Java et propose une édition *open source* ;
- SaxonCS sur la plateforme .NET n'offre qu'une version entreprise ;
- SaxonC est en C/C++ et propose une édition *open source* ;
- SaxonJS est écrit en JavaScript et fonctionne dans l'environnement Node.js ainsi que dans la plupart des navigateurs récents permettant ainsi une intégration du côté du web.

Contraintes

En regard de la demande formulée par l'équipe commanditaire de ce rapport, nous notons que l'une des principales contraintes à propos de Saxon est qu'une large partie des éditions que Saxonica propose ne sont pas *open source* et sont la propriété de l'entreprise. C'est notamment le cas pour SaxonJS, le module dédié à l'intégration web qui nous intéresse le plus pour la constitution d'une forge. L'autre contrainte à déplorer est la distinction des communautés entre les utilisateurs des générateurs de site statique et les utilisateurs de processeurs XML. En conséquence, il n'existe que très peu

de ponts à tisser entre un format XML comme source primaire, soit en entrée du processus, et une forge basée sur le principe du *Single Source Publishing*.

SINGLE SOURCE PUBLISHING

Le concept de *Single Source Publishing*, traduit par Publication à partir d'une source unique, désigne le fait de générer plusieurs formats à partir d'une seule et même source (Hyde, 2021). Un seul et unique document permet de produire des formats divers, sans avoir besoin de basculer d'une version de travail à une autre. Que ce soit un format PDF pour l'impression, un export XML pour un diffuseur numérique ou une version numérique au format HTML, l'objectif est de travailler avec une même source.

Cette source n'est pas forcément un seul fichier, mais peut comprendre des fichiers textes, des *médias* comme des images, des fichiers son ou des vidéos, mais aussi des données structurées de différentes façons. Il s'agit d'un enjeu éditorial qui soulève des questions autant théoriques que techniques, telles que la légitimation des contenus, l'évolution des pratiques d'édition ou la création d'outils adéquats, que nous ne détaillerons pas ici.

Le *Single Source Publishing* est une problématique autant pour les éditeurs que pour les distributeurs/diffuseurs, car il permet de produire autant de formats que nécessaire tout en conservant une même origine. Les apports sont nombreux :

- création d'un espace commun pour tous les acteurs de la chaîne d'édition ;
- clarification des interventions sur les contenus ;
- croisement des besoins entre les différents formats de sortie ;
- mutualisation des efforts pour les différents exports à produire (PDF, HTML/web, XML, EPUB, etc.) ;
- simplification de l'archivage en ne disposant que d'une seule et même source.

La mise en place de chaînes de publication implémentant le *Single Source Publishing* est complexe, plusieurs initiatives tentent de réaliser ce défi depuis les débuts de l'informatique. Les implémentations sont nombreuses, et respectent de façon plus ou moins approfondie les contraintes imposées par cette approche : LaTeX, Métopes, Stylo, Quire, Manifold, etc.

SITE WEB COMPLEXE

Un site web complexe sera dissocié d'une forme plus simple comme le blog, dont la vocation est d'afficher des contenus, selon les fonctionnalités qu'il présente et qui consomment plus de ressources : gestion des contenus multi-pages, calculs à effectuer côté serveur, gestion de base de données, etc.

SVELTE & SVELTEKIT

Svelte est une librairie de composant JavaScript, comme React, et SvelteKit est un *framework* pour la construction d'interface utilisateur basé sur Svelte, très similaire à son homologue Next.js.

L'approche de Svelte est tout à fait novatrice vis-à-vis de ses concurrents : Svelte propose une amélioration des performances de construction des applications (lors de l'exécution du *build*) par l'utilisation d'un compilateur plutôt que de la traditionnelle différenciation dans le DOM virtuel. Cette composante nous permet d'observer une nouvelle évolution dans la lignée des applications web, comme les générateurs de site statique, renouvellant ainsi une dynamique qui paraissait s'essouffler depuis quelques temps.

À l'instar des autres technologies de la nouvelle génération de librairies et de *frameworks* pour construire des applications web, Svelte et SvelteKit mobilisent de nouveaux concepts techniques pour améliorer les performances de ces dernières.

SYSTÈME DE GESTION DES CONTENUS (CMS)

CMS est l'acronyme de *Content Management System*, en français Système de Gestion de Contenu, et fait référence aux logiciels de gestion et d'administration des sites web et de leurs contenus stockés sur des bases de données. Le CMS se présente généralement sous la forme d'une interface, appelée *back-office*, accessible uniquement pour qui en a les identifiants, et permet d'administrer complètement un site web : de la gestion des pages à la gestion des différents modules implémentés.

TEXINFO

TexInfo est le format de documentation officiel du projet GNU. Il a été conçu en 1986 par Richard Stallman et Robert Chassel. Plus précisément, TexInfo est un langage de formatage de texte qui s'appuie sur les technologies *Tex* pour convertir une source unique en différents formats destinés au web ou à l'impression.

TINA

Tina est un gestionnaire de contenus (CMS), tout comme Forestry dont il est issu. Il offre la possibilité d'administrer le contenu d'un site web statique depuis une interface graphique. Cette application très légère comprend :

- un éditeur de texte en Markdown ;
- une connexion à Git grâce à l'API (permettant ainsi la mise à jour des contenus directement depuis l'interface graphique) ;
- la possibilité d'éditer directement les contenus depuis la page Web du site.

TRANSCRIPTION (ESCRIPTORIUM)

Description générale

eScriptorium est une application web adossée au moteur d'HTR (*Handwritten text recognition*) Kraken. Elle est principalement développée par le projet SCRIPTA (PSL) mais, étant un projet entièrement ouvert, elle a reçu et reçoit toujours des contributions de la part de plusieurs partenaires sous la forme de développement de fonctionnalités ou d'éléments de documentation, notamment de la part d'Inria (ALMAnaCH) et de l'Université du Maryland (OpenITI).

Exemples d'utilisation

Deux types d'exemples peuvent être présentés :

- Sans moteur d'HTR, comme plateforme ergonomique de transcription (notamment pour la création de données d'entraînement) ;
- Via moteur d'HTR, pour des projets qui visent :
    - à rendre accessibles des fonds patrimoniaux (cf. LECTAUREP) ;
    - à créer des éditions numériques à partir du texte produit automatiquement (cf. e-Notre-Dame-de-Paris) ;
    - à créer des données sur lesquels appliquer des outils d'analyse de texte.

Contraintes

Deux principales contraintes sont remarquables à propos d'eScriptorium :

- tout d'abord, les développements sont toujours en cours. En conséquences, la documentation n'est pas forcément à jour vis-à-vis de ces derniers et certaines fonctionnalités ne sont pas encore déployées (ex : alignement automatique du texte avec les images, *a posteriori* ; ou encore l'export TEI XML à peu près standard)
- suppose d'avoir soit des capacités de calcul propre (et des compétences en administration système si on veut créer une infrastructure pour un laboratoire ou un groupe de collègues) soit d'avoir accès à un serveur maintenu par une autre infrastructure (ex : serveur Inria / CREMMA).

WEB COMPONENT

Les composants web (*Web Components*) sont un ensemble de plusieurs technologies qui permettent de créer des éléments personnalisés réutilisables, dont les fonctionnalités sont encapsulées en dehors du reste du code et qui peuvent être utilisées au sein d'applications web.

WEB SÉMANTIQUE

Les technologies du web sémantique et l'approche du web de données liées désignent une combinaison de techniques, d'outils et de standards qui permettent de transformer le World Wide Web d'un web de documents à un web de données. Lorsque cette approche est appliquée au monde des bibliothèques, des archives et des musées, les données liées transforment la manière dont nous pouvons découvrir, analyser, et visualiser les contenus culturels et scientifiques. Les données ouvertes et liées (*Linked Open Data*, LOD) permettent aux institutions patrimoniales et culturelles de publier et de partager des informations sur leur collections en ouvrant d'infinies possibilités de réutilisations et d'enrichissements et afin d'augmenter leur visibilité.

Le web sémantique repose sur différentes technologies dont le RDF (*Resource Description Framework*, formalisation et description des objets en triplets), des ontologies et des vocabulaires pour définir les relations entre les objets, et SPARQL, un langage de requête dédié aux triplets RDF.

XML

XML pour *eXtensible Markup Language*; est également un métalangage de balisage et de modélisation du texte. Plus souple que le HTML dont les balises sont figées, XML permet à chaque utilisateur de créer son propre système hiérarchique arborescent par l'élabor-

ation de balises personnalisées. Postérieur d'une décennie au HTML, la publication des recommandations de la première version (1.0) du métalangage XML voit le jour en 1998.

La description rigoureuse permise grâce à cette technologie en fait un outil utilisé à plusieurs fins notamment l'élaboration d'éditions critiques de certains textes, qu'ils soient anciens ou nativement numériques ou encore la description formelle de jeux de données (jusqu'à la création de bases de données). XML peut être associé à un autre langage, le XSL (eXtensible Stylesheet Language), qui décrit comment doit être transformé le XML.

XML est un langage supporté par les navigateurs web et est facilement transformable en HTML et compatible avec le CSS.

YAML

YAML dans sa version originale de 2004 est l'acronyme de *Yet Another Markup Language* puis se transforme à l'occasion de la publication de sa version 1.1 en *YAML Ain't Markup Language*. YAML est un langage de sérialisation de données pour tous les langage de programmation. Dans le cas des outils liés à l'édition numérique, YAML sera utilisé pour enregistrer les métadonnées associées à un document.

ÉCRITURE

L'écriture comprend tout d'abord une phase d'inscription. Elle consiste en l'action d'écriture de l'information sur le support matériel - *hardware* - que réalise la machine. C'est un paramètre qu'il ne faut pas oublier, l'écriture n'est à aucun moment située dans le logiciel mais bien dans le matériel physique qui stocke les informations (Kittler, 2015). Toutes les opérations qui interviennent par l'entremise des logiciels et des forges ne sont que des suites d'ordres et de commandes envoyées à la machine pour qu'elle procède à cette inscription. Ainsi, toute action réalisée dans le champ du numérique s'apparenterait à une forme d'écriture. Écrire ne relève plus uniquement du geste de tracer des lignes mais devient une application plus globale de l'éditorialisation (Vitali-Rosati, 2016), dans laquelle interviennent des algorithmes, des logiciels,

des interfaces, des robots, etc. Chaque événement — *capta* — est inscrit dans une couche différente, dans un paratexte numérique (Genette, 1982; Vitali-Rosati, 2020) qui prend la forme d'un palimpseste (Kembellec, 2020; Mellet, 2020) dont les délimitations permettent l'apparition du document numérique. Le processus d'écriture est donc constant, un pic d'activité est remarquable en début de la chaîne d'une forge, lorsqu'il s'agit de générer les documents à publier : écriture du texte, des images, des sons.

COLOPHON

Ce rapport a été rédigé entre février 2022 et août 2023 avant d'être déposé en décembre 2024.

Commande du rapport : Emmanuel Château-Dutier (CIÉCO), Michael Sinatra (CRIHN).

Projet coordonné et écrit par Roch Delannay et Antoine Fauchié, tous les deux doctorants à l'Université de Montréal au moment de la rédaction.

Ce document est produit avec Markdown, 11ty, Git, GitLab, Paged.js, Vim, VSCodium et Ubuntu.