Filière Informatique de gestion
Travail de Bachelor

Browser RERO DOC

2018

Prof. Dominique Genoud
Responsables RERO : Miguel Moreira, Johnny Mariéthoz

Christian Pereira

Résumé

L'objectif de ce travail est de remplacer le navigateur existant sur www.doc.rero.ch qui permet de consulter documents PDF, des images tels que des thèses, des livres, etc.

Pour atteindre cet objectif, l'état de l'art des solutions pour la manipulation d'images et des PDF a été réalisé.

Pendant l'analyse, des problèmes ont été rencontrés pour l'extraction de texte de fichier PDF. Les librairies écrites en python n'étaient pas satisfaisantes pour les objectifs de notre application.

Une fois la solution trouvée, nous avons mis en place le serveur et choisi la technologie pour notre frontend.

A ce jour, un prototype de ce navigateur est disponible et permet une navigation parmi les documents.

Mots-clés : RERO, Angular, Python, Cython, Invenio, Navigateur multimédia

Avant-propos et remerciements

Ce travail a été réalisé dans le cadre du projet de Bachelor de la filière informatique de gestion HES-SO Valais Wallis.

Il a été proposé par le RERO (réseau des bibliothèques de la Suisse occidentale) (RERO, 2017) et suivi par le professeur, Dominique Genoud. Le RERO a pour objectif la réalisation d'une interface utilisateur pour la consultation des documents numériques présents sur la bibliothèque numérique RERO DOC (RERO, 2005). A ce jour, RERO dispose déjà d'une solution pour la consultation de ces documents. Ce produit datant de 2011 n'est plus vraiment adapté, c'est pour cela que le RERO a proposé ce travail afin qu'une étude sur les solutions possibles puisse être réalisée.

Pour atteindre ce but, j'ai collaboré avec Johnny Mariéthoz de manière à satisfaire les objectifs demandés.

Remerciements

J'aimerais remercier toutes les personnes qui ont contribué à l'élaboration de ce travail.

Un remerciement particulier à :

- Johnny Mariéthoz, développeur confirmé de l'équipe du RERO, pour son suivi et pour ses conseils tout au long du projet.
- Dominique Genoud, professeur à l'HES, pour le suivi de mon Bachelor et ses conseils lors de nos séances.
- Catia Da Silva, ma compagne, pour la relecture du document et son soutien tout au long de cette formation.

TABLE DES MATIÈRES

LISTE DES ILLUSTRATIONS

LISTE DES TABLEAUX

Glossaire

Apache 2.0 :

La licence Apache est une licence de logiciel libre et open source. Elle est écrite par l'Apache Software Foundation, qui l'applique à tous les logiciels qu'elle publie. Il existe plusieurs versions de cette licence (1.0, 1.1, 2.0). Cette licence n'est pas copyleft. (Licence Apache, 2018)

AVX2 :

Advanced Vector Extensions (AVX) est un jeu d'instructions de l'architecture x86 d'Intel et AMD. (Advanced Vector Extensions, 2018)

Cython :

Cython est un langage de programmation et un compilateur qui simplifient l'écriture d'extensions compilées pour Python. (Cython, 2018)

DPI :

Le point par pouce (PPP, ppp, DPI ou dpi) est une unité de précision communément utilisée pour définir la résolution d'un scanner, d'une imprimante ou d'une souris optique. (Point par pouce, 2018)

JSX :

JavaScript Syntax eXtension : extension du langage de programmation JavaScript.

Licence BSD :

La licence BSD (Berkeley Software Distribution License) est une licence libre utilisée pour la distribution de logiciels. (Licence BSD, 2017)

Licence GNU GPL :

La licence publique générale GNU, ou GNU General Public License (son seul nom officiel en anglais, communément abrégé GNU GPL, voire simplement « GPL »), est une licence qui fixe les conditions légales de distribution d'un logiciel libre du projet GNU. (Licence publique générale GNU, 2018)

Licence MIT :

C'est une licence de logiciel libre et open source, non copyleft, permettant donc d'inclure des modifications sous d'autres licences, y compris non libres. (Licence MIT, 2018)

OOP :

Programmation orienté objet.

RERO :

Réseau romand des bibliothèques de Suisse occidentale. (RERO, 2017)

SIMD :

Single Instruction on Multiple Data, ou SIMD, est une des quatre catégories d'architecture définies par la taxonomie de Flynn en 1966 et désigne un mode de fonctionnement des ordinateurs dotés de capacités de parallélisme. (Single instruction multiple data, 2018)

TypeScript :

TypeScript est un langage de programmation libre et open source développée par Microsoft qui a pour but d'améliorer et de sécuriser la production de code JavaScript. (TypeScript — Wikipédia, 2018)

Wrapper :

En génie logiciel, adaptateur ou wrapper est un patron de conception de type structure. Il permet de convertir l'interface d'une classe en une autre interface que le client attend. L'adaptateur fait fonctionner ensemble des classes qui n'auraient pas pu fonctionner sans lui, à cause d'une incompatibilité d'interfaces. (Adaptateur (patron de conception), 2018)

Introduction

La consultation de fichiers PDF ou d'autres types de documents numériques implique aux navigateurs web d'aujourd'hui de télécharger ces éléments avant de pouvoir les consulter.

Selon la taille de ces mêmes documents, cela peut rapidement devenir un problème car certains documents peuvent atteindre des tailles importantes (supérieures à 100 MB). Pour les appareils mobiles, cela devient vite problématique car leur espace mémoire est limitée et selon la vitesse de connexion disponible, le temp de chargement sera long.

Le RERO, réseau des bibliothèques de la Suisse occidentale, met à disposition sur son site (www.doc.rero.ch) la possibilité de consulter des documents littéraires scientifiques comme des articles, des thèses ou mêmes des enregistrements sonores. Ces documents doivent pouvoir être consultés de manière rapide ; sans avoir besoin de télécharger l'intégralité du document.

Pour rendre cela possible, un serveur a été mis en place ainsi qu'une interface web permettant aux visiteurs de consulter ces documents. Cette solution s'appelle Multivio et elle est disponible à l'adresse suivante www.multivio.org. Cette solution datée de 2011 est encore fonctionnelle mais rencontre des problèmes de stabilités et est basée sur une technologie obsolète. Multivio est devenu au cours du temps difficile à maintenir. C'est pour cela que le RERO cherche une solution basée sur des technologies à l'état de l'art.

Ce développement open source impose que le serveur doit être écrit en python sous forme d'un module Invenio. Invenio est un framework open source pour les référentiels numériques à grande échelle. Ce framework utilise les meilleurs outils open source pour que la bibliothèque numérique soit évolutive, sûre et rapide. Développé initialement par le CERN, il bénéficie aujourd'hui aussi de contributions externes.

Dans ce rapport, nous allons dans un premier temps faire une analyse de la solution existante pour comprendre ces actuels défauts et qualités.

Ensuite, dans la deuxième partie du projet, nous allons rechercher les nouvelles technologies pour établir l'état de l'art des solutions existantes afin de remédier aux problèmes rencontrés dans l'analyse précédente.

Pendant toute l'analyse du projet, nous devrons aussi tenir compte que les projets RERO sont actuellement sous licence GPLv2. Si un problème de compatibilité venait à apparaitre nous pourrions aussi utiliser la licence GPLv3.

Pour terminer, la solution existante sera détaillée afin de proposer une meilleure alternative.

1. Analyse des solutions existantes

1.1. Fonctionnement

Pour que le navigateur soit fonctionnel, le RERO a mis en place un serveur dédié aux traitements des documents et une application web pour l'interface utilisateur.

Voici la description de son fonctionnement :

Dès qu'une publication est mise à disposition sur leur site, un fichier XML est généré et importé sur le serveur du RERO. Ce fichier est fondamental car dans celui-ci, nous retrouvons les informations relatives aux documents faisant partie de cette publication. De plus, le navigateur du RERO permet de mixer les types des fichiers dans une même publication, par exemple une thèse avec un fichier principal en PDF avec des annexes en format JPEG.

Le fichier XML contient, en plus des informations concernant les documents, leur emplacement et leur type ainsi que des informations générales sur la publication tel que le titre.

Ceci est le point de départ pour notre application. Lorsque nous ouvrons le navigateur, les informations vont être chargées et ainsi nous saurons quels documents nous devrons afficher.

Un exemple de ce fichier est disponible à l'annexe I.

En image le fonctionnement décrit ci-dessus :

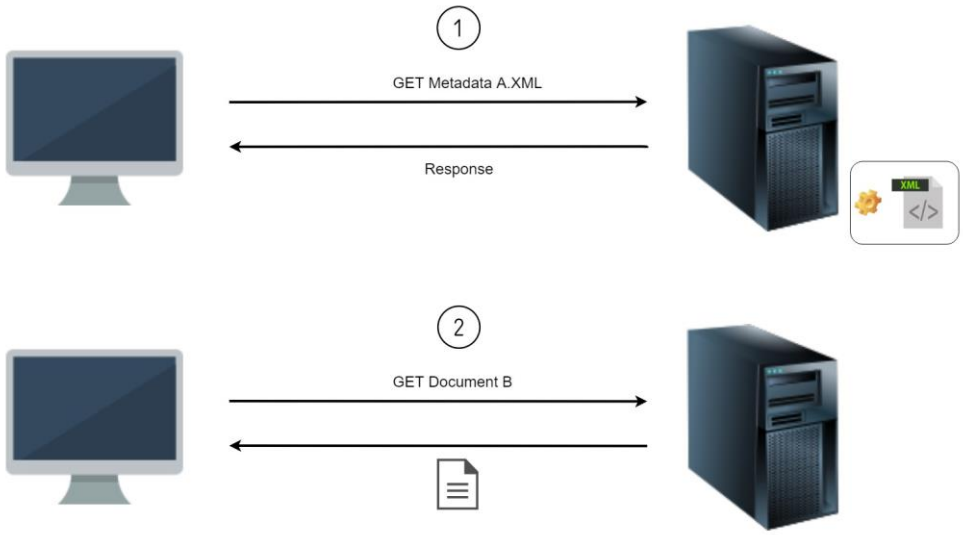

Figure 1: Fonctionnement de la solution existante

Une dernière caractéristique de la solution actuelle est que lorsque que nous visionnons un document PDF, le serveur transforme la page demandée en image JPEG. Cette technique nous évite par conséquent de télécharger l'intégralité du fichier.

1.2. Backend

Le serveur est écrit en Python version 2. Autre la logique du serveur pour desservir les requêtes pour l'interface graphique, son axe principal est l'utilisation de Swig et Poppler pour le rendu des fichiers PDF, la recherche et l'extraction de texte.

1.2.1. Swig

Swig est un logiciel open source qui permet à des langages comme Python de communiquer avec des autres bibliothèques écrites en C/C++. Swig est compatible avec la version de Python 3 mais la version conseillée pour son utilisation est la version 2.3.

Swig va permettre une conversion des types afin que la communication entre les deux langages soit possible.

1.2.2. Poppler

Poppler est une librairie écrite en C++ pour l'extraction de textes/images depuis une source de type PDF.

Cette librairie est toujours maintenue est comporte des bonnes performances pour la manipulation des fichiers PDF.

1.2.3. Problèmes

Cet interfaçage entre Swig et Poppler a bien fonctionné à ses débuts mais depuis la version 0.18 de Poppler (15 février 2012), la liaison ne se fait plus correctement.

Actuellement, nous sommes à la version 0.67 de Poppler. Ceci est un problème car il ne permet pas aux développeurs d'utiliser les dernières mises à jour de leur logiciel clé.

De plus, l'équipe du RERO aimerait pouvoir mettre à jour leur version de Python sur le serveur en passant à la version 3 mais comme nous avons vu dans le chapitre sur Swig, même s'il est compatible avec les versions 3 de Python, ceci n'est pas la meilleure combinaison pour cet outil.

Pour terminer, l'équipe du RERO utilise maintenant les modules Invenio (framework développé par le CERN) qui depuis sa version 3 déconseille l'utilisation de la version 2 de Python. L'utilisation

de ces modules est une des contraintes du RERO car leur objectif final est l'implémentation du backend sous forme de module Invenio.

Tous ces points, on conduit RERO à repenser leur serveur pour le rendre plus stable et plus fiable.

1.3. Frontend (Browser)

Cette application écrite en JavaScript avec le framework SproutCore s'exécute dans la page web et permet de visualiser les documents PDF et d'autres documents sans avoir besoin de les télécharger.

Même si cette solution est fonctionnelle, elle comporte des problèmes au niveau de son fonctionnement et de sa stabilité.

1.3.1. Problèmes

L'utilisation de SproutCore 1.6 représente le principal problème pour cette solution car cette technologie n'est plus supportée donc plus maintenable. De plus, sa stabilité est parfois faible et après une analyse approfondie de son fonctionnement, on constate que certaines requêtes s'exécutent plusieurs fois. Ceci peut être rapidement repéré lors du chargement d'un document. Cela charge inutilement le serveur est baisse les performances de l'application. Ce frontend supporte d'anciens formats tels que MODS et MEDS, qui ne sont plus utilisés par RERO. Ceci apportait une complexité plus nécessaire qui devrait simplifier son implémentation.

Pour résoudre ces problèmes, il va falloir trouver un nouvel framework JavaScript afin de pouvoir développer un nouveau navigateur et l'intégrer dans le site existant sans avoir besoin de refaire le site en entier.

2. Compatibilité licences

Ci-dessous le récapitulatif concernant la compatibilité des licences avec GPLv3.

Cette figure nous permettra tout au long de nos analyses de déterminer si la compatibilité est conforme.

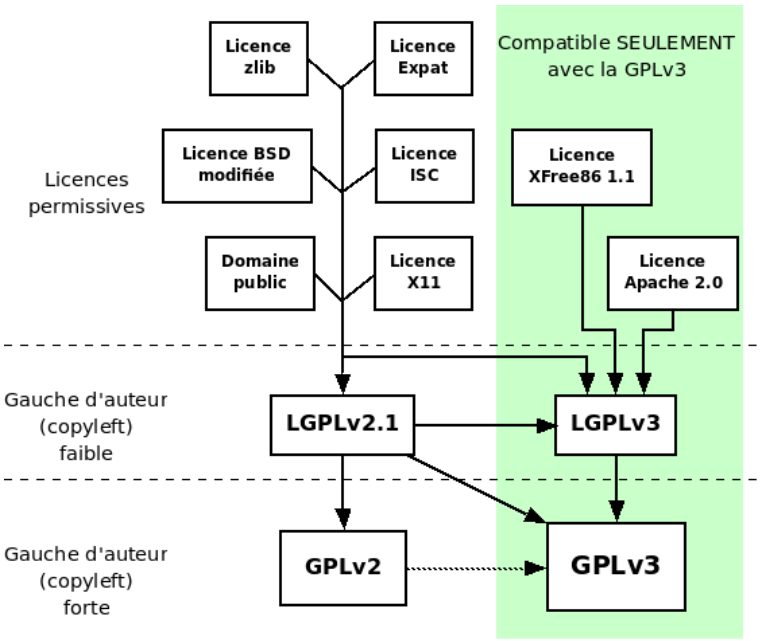

Figure 2: Récapitulatif de la compatibilité des licences (Smith, 2016)

Les flèches pointant d'une licence à une autre indiquent que la première licence est compatible avec la seconde. Ceci est vrai même en suivant plusieurs flèches à la suite ; donc, par exemple, la licence ISC est compatible avec la GPLv3. (Smith, 2016)

3. Backend

Dans ce premier chapitre, nous allons analyser les solutions disponibles à ce jour permettant de répondre aux problèmes détectés dans le premier chapitre.

Dans cette analyse, les solutions de type PDFJs ne sont pas prises en compte car même si elles sont très utilisées pour l'affichage des documents dans les pages web, ces librairies ne sont pas acceptables pour notre cas de figure. Nous avons besoin d'un outil capable de manipuler des gros fichiers PDF ce qui est plus difficile dans ce type de librairies. De plus, les manipulations des documents seront effectuées côté serveur qui est écrit en Python.

3.1. Solutions imposées

Les contraintes imposées par le RERO pour le nouveau serveur sont les suivantes :

- Développement en Python 3
- Backend sous la forme d'un module Invenio
- Projet open source

3.2. Environnement tests

Chaque test réalisé lors de l'analyse sera effectué dans un environnement Python isolé (virtualenv). Ainsi, les tests seront indépendants du système et ses librairies.

Les informations concernant le système sont disponibles à l'annexe II.

3.3. Manipulations d'images

3.3.1. Etat de l'art

Un des premiers objectifs de notre navigateur est de pouvoir visualiser et manipuler les images.

Après une analyse, les librairies retenues sont :

- Pillow, Pillow-SIMD, OpenCV et Wand.

Pour déterminer quelle librairie on va retenir, des tests sur le temps d'exécution pour une même tâche ont été mesurés. Ainsi, nous pourrons choisir quelle librairie est la plus adaptée à notre application.

Pillow

Pillow est un fork de la librairie PIL (Python Imaging Library), projet de type open source qui est actuellement la librairie utilisée par le serveur du RERO.

Licence du projet, PIL Software License.

Pillow-SIMD

Pillow-SIMD n'est outre qu'une autre version de Pillow qui a comme objectif d'améliorer ses performances grâce à l'utilisation des instructions SIMD (Single Instruction Multiple Data).

Parfaitement compatible avec Pillow, cela permet de pouvoir passer d'une librairie à l'autre sans avoir besoin de modifier son code.

Comme pour Pillow, de licence PIL Software License.

OpenCV

OpenCV (Open Computer Vision) est une librairie graphique libre spécialisée dans le traitement d'images en temps réel.

Projet sous licence 3-clause BSD License.

Wand

Wrapper de la librairie ImageMagick, Wand utilise les fonctions natives en C d'ImageMagick sous Python.

Wand est un projet open source sous licence MIT.

Pillow	Pillow-SIMD	OpenCV	Wand
✓	✓	✓	✓

Tableau 1: Analyse manipulations d'images, compatibilité des licences

3.3.2. Comparaisons des solutions existantes

Pillow, OpenCV et Wand

Comme nous ne pouvons pas utiliser Pillow et Pillow-SIMD en même temps lors des tests, nous allons utiliser Pillow dans un premier temps. Ensuite, le gagnant de ce premier test sera mis en confrontation avec Pillow-SIMD.

Pour commencer, nous allons effectuer un chargement d'image, une rotation, une transposition et une modification de taille. Pour cet essai, nous avons utilisé quatre images de tailles différentes. Cette opération sera exécutée en une seule itération puis dans un deuxième temps, ce même test sera fait avec dix itérations à la suite afin de voir la capacité de charge de ces outils. Le script de ce test est disponible dans l'annexe III suivi des tableaux avec les mesures d'exécution.

Suite aux mesures reportés dans l'annexe, nous constatons que OpenCV remporte légèrement ce premier test mais sur des images de plus grande taille, Pillow obtient aussi des bons résultats. En revanche, Wand est le grand perdant de ce premier test. Voici nos évaluations suite à ce premier test :

Pillow	OpenCV	Wand
3,5/5	4/5	2/5

Tableau 2: Evaluation des librairies, test 1

Maintenant, nous allons confronter Pillow-SIMD à OpenCV car celle-ci est proclamée par ses auteurs la librairie plus rapide sous Python.

Pillow-SIMD, OpenCV

Le script de test reste invarié. L'unique différence est que maintenant nous utilisons Pillow-SIMD au lieu de Pillow (code disponible à l'annexe IV, suivi des tableaux des mesures). Pour évaluer encore plus en détails les performances de ces librairies, nous avons additionné toutes les opérations afin de pouvoir les départager et les comparer.

Les performances sont encore très proches mais dès qu'on travaille avec des images de taille plus importantes et avec un nombre d'itérations plus élevé, Pillow-SIMD devient nettement plus performant avec des différences de temps non négligeables.

Pillow	OpenCV
4,5/5	4/5

Tableau 3: Evaluation des librairies, test 2

Pillow SIMD performance test

L'équipe de Pillow-SIMD fournit aussi un outil pour pouvoir comparer leur librairie avec celles de la concurrence.

Dans ce test, nous pouvons choisir quelles fonctions tester et quelles librairies confronter à Pillow-SIMD.

Les fonctions plus intéressantes pour notre serveur ont été comparées :

```
Load 2560×1600 RGB image
    Jpeg load           0.07657 s    53.49 Mpx/s
    Jpeg save           0.10354 s    39.56 Mpx/s

Cv2 load 2560×1600 RGB image
    Jpeg load           0.10977 s    37.32 Mpx/s
    Jpeg save           0.10862 s    37.71 Mpx/s

Rotate right 2560×1600 RGB image
    Flop                0.01050 s   390.13 Mpx/s
    Flip                0.00432 s   947.70 Mpx/s
    Rotate 90           0.01747 s   234.47 Mpx/s
    Rotate 180          0.01056 s   387.99 Mpx/s
    Rotate 270          0.01607 s   254.90 Mpx/s
    Transpose           0.01587 s   258.13 Mpx/s
    Transverse          0.01784 s   229.65 Mpx/s

Cv2 rotate right 2560×1600 RGB image
    Flop                0.02413 s   169.78 Mpx/s
    Flip                0.00466 s   878.95 Mpx/s
    Rotate 90           0.02714 s   150.90 Mpx/s
    Rotate 180          0.02493 s   164.28 Mpx/s
    Rotate 270          0.04983 s    82.19 Mpx/s
    Transpose           0.02517 s   162.75 Mpx/s
    Transverse          0.05287 s    77.48 Mpx/s

Scale 2560×1600 RGB image
    to 26x16 bil        0.00886 s   462.36 Mpx/s
    to 26x16 bic        0.01664 s   246.19 Mpx/s
    to 26x16 lzs        0.02838 s   144.32 Mpx/s
    to 320x200 bil      0.01143 s   358.48 Mpx/s
    to 320x200 bic      0.02048 s   199.96 Mpx/s
    to 320x200 lzs      0.03261 s   125.61 Mpx/s
    to 2048x1280 bil    0.04350 s    94.16 Mpx/s
    to 2048x1280 bic    0.06239 s    65.65 Mpx/s
    to 2048x1280 lzs    0.08232 s    49.76 Mpx/s
    to 5478x3424 bil    0.26307 s    15.57 Mpx/s
    to 5478x3424 bic    0.33774 s    12.13 Mpx/s
    to 5478x3424 lzs    0.41726 s     9.82 Mpx/s

Cv2 scale 2560×1600 RGB image
    to 26x16 sup        0.06288 s    65.14 Mpx/s
    to 320x200 sup      0.02065 s   198.31 Mpx/s
    to 2048x1280 sup    0.20232 s    20.24 Mpx/s
    to 5478x3424 bil    0.24546 s    16.69 Mpx/s
    to 5478x3424 bic    0.44551 s     9.19 Mpx/s
    to 5478x3424 lzs4   1.54130 s     2.66 Mpx/s
```

Figure 3: Pillow-SIMD, test de performance

Ci-dessus, Pillow-SIMD remporte le match amplement. Ce test fournit par l'équipe de Pillow-SIMD est à interpréter avec précaution car nous ne connaissons pas l'exactitude de son déroulement.

3.3.3. Choix

On serait tenté d'utiliser OpenCV car il a remporté la majorité des tests. Cependant, du point de vue de notre application, les méthodes le plus sollicitées sont le chargement et le redimensionnement. Dans ce contexte, lors d'une grosse charge, Pillow-SIMD se comporte nettement mieux. De plus, sur son site, le RERO permet d'ajouter d'anciennes cartes géographiques qui ont souvent une haute résolution et suite à nos tests, nous savons maintenant que Pillow-SIMD est aussi nettement plus performant dans ce cas de figure.

Concernant l'utilisation, les deux outils s'installent aisément mais Pillow-SIMD a une syntaxe plus simple et plus facile à lire.

Selon les besoins, Pillow-SIMD peut être recompilé pour des processeurs supportant la technologie AVX2 qui permet d'augmenter ses performances.

Pour comparatifs, les nouvelles mesures de Pillow-SIMD avec support AVX2.

```
Load 2560×1600 RGB image
    Jpeg load           0.07705 s    53.16 Mpx/s
    Jpeg save           0.10357 s    39.55 Mpx/s

Rotate right 2560×1600 RGB image
    Flop                0.00935 s   438.31 Mpx/s
    Flip                0.00438 s   934.70 Mpx/s
    Rotate 90           0.01743 s   235.05 Mpx/s
    Rotate 180          0.00927 s   441.62 Mpx/s
    Rotate 270          0.01582 s   258.95 Mpx/s
    Transpose           0.01569 s   261.06 Mpx/s
    Transverse          0.01774 s   230.94 Mpx/s

Scale 2560×1600 RGB image
    to 26x16 bil        0.00554 s   739.49 Mpx/s
    to 26x16 bic        0.01016 s   403.19 Mpx/s
    to 26x16 lzs        0.01891 s   216.65 Mpx/s
    to 320x200 bil      0.00797 s   513.94 Mpx/s
    to 320x200 bic      0.01320 s   310.21 Mpx/s
    to 320x200 lzs      0.02212 s   185.21 Mpx/s
    to 2048x1280 bil    0.03231 s   126.79 Mpx/s
    to 2048x1280 bic    0.04362 s    93.89 Mpx/s
    to 2048x1280 lzs    0.05712 s    71.71 Mpx/s
    to 5478x3424 bil    0.22074 s    18.56 Mpx/s
    to 5478x3424 bic    0.26830 s    15.27 Mpx/s
    to 5478x3424 lzs    0.33345 s    12.28 Mpx/s
```

Figure 4: Pillow-SIMD, test de performance avec AVX2

C'est pour toutes ces raisons que le choix se porte sur Pillow-SIMD.

3.4. Rendu des fichiers PDF

3.4.1. Etat de l'art

La deuxième analyse concerne le rendu d'images depuis un PDF. Cette technique est essentielle car notre serveur doit être capable de transformer une page spécifique du PDF en image. Sans cela, notre solution perdrait son intérêt vis-à-vis de la consultation des documents PDF.

À la suite des recherches, les librairies retenues sont les suivantes :

- Wand, Pdf2image et Pdftoppm

Wand

Wrapper de la librairie ImageMagick, Wand utilise les fonctions natives en C d'ImageMagick sous Python.

Wand est un projet sous licence MIT, compatible avec notre projet.

Pdftoppm

Pdftoppm est un outil fourni par Poppler qui permet d'utiliser les fonctions de conversion de Poppler via la ligne de commandes.

De licence GPLv2 elle est compatible avec le projet.

Pdf2image

Pdf2image est un module qui enveloppe pdftoppm. Le but de cette librairie est de simplifier l'utilisation de pdftoppm.

Pdf2image est un projet sous licence MIT.

Wand	Pdftoppm	Pdf2image
✓	✓	✓

Tableau 4: Analyse du rendu des fichiers PDF, compatibilité des licences

3.4.2. Comparaison des solutions existantes

Pour ce test, nous essayerons de convertir une page spécifique du PDF. Le résultat final sera une image JPEG avec une résolution de 300dpi. Comme lors de la première analyse, on va mesurer quel outil va générer plus rapidement notre image (script annexe V).

En image le résultat du test :

```
------------ PDF conversion tests ----------

---- pdf2image ----
Elapsed time: 0.908712

---- wand ----
Elapsed time: 0.581810

---- pdftoppm ----
Elapsed time: 0.095475
```

Figure 5: Résultat du temps d'exécution depuis un PDF vers une image

Le résultat montre clairement que pdftoppm est nettement plus rapide pour ce type de tâche.

3.4.3. Choix

Cette analyse se résout assez facilement car pdftoppm (Poppler) même s'il est moins facile à utiliser en vue de sa syntaxe, il ne rencontre pas d'adversaire à sa taille lors du comparatif des temps d'exécution.

Pdf2Image englobe assez bien pdftoppm mais le manque de possibilité de choisir la page à convertir rend cet outil impossible à utiliser.

Pour rappel dans notre analyse, nous n'avons fait aucun test qualitatif du rendu par manque de mesure objective.

3.5. Extraction du texte contenu dans un fichier PDF

3.5.1. Etat de l'art

Cette analyse a pour but de trouver une librairie capable d'extraire des informations telles que le texte de contenu, les métadonnées ou la table des matières depuis une source PDF.

Cette partie est sans aucun doute l'analyse la plus pertinente. L'application doit être capable de trouver le plus d'informations possibles dans les fichiers PDF. La table des matières, par exemple, devra contenir l'information de la page de destination pour chaque élément la composant. Une des fonctionnalités de notre navigateur consiste en effet à pouvoir rediriger l'utilisateur depuis la table des matières vers la page recherchée.

Après différentes études, voici les librairies qui ont été retenues :

- Slate, pypd2f, pdfminer3k

Slate

Ce package est basé sur Pdfminer. Son but principal est de simplifier l'extraction de texte depuis un fichier PDF puisque Pdfminer est très complexe et très vaste. Cette librairie est en version bêta.

License du projet, MIT.

Pypdf2

Cette librairie écrite en Python permet d'extraire les informations du PDF et son texte. Elle permet également de fusionner un ensemble de fichiers PDF.

Licence du projet BSD.

Pdfminer3k

Cette librairie n'est que la version pour Python 3 de Pdfminer. Elle permet d'extraire les informations relatives aux PDF et lors de la recherche de texte, elle permet de les localiser.

Licence du projet, MIT.

Slate	Pypdf2	Pdfminer3k
✓	✓	✓

Tableau 5: Analyse d'extraction du texte contenu dans un PDF, compatibilité des licences

3.5.2. Comparaisons des solutions existantes

Extraction texte

Pour ce test, nous allons simplement extraire du texte d'une page PDF. Le temps d'exécution sera encore mesuré. Ce test a été réalisé sur un petit échantillon de documents.

Le script de ce test est disponible à l'annexe VI.

Résultat d'exécution :

```
(OCR) christian@ubuntu-lenovo:~/virtenvs/OCR$ python script.py

------------ OCR ----------

---- slate ----
Elapsed time: 0.929086

---- pypdf2 ----
File Decrypted (PyPDF2)
Elapsed time: 0.019573

---- pdfminer ----
Elapsed time: 0.077469
```

Figure 6: Résultat du temps d'exécution pour l'extraction du texte

Slate est très vite écarté par ses concurrents car son temps d'exécution est très mauvais. La suite de la comparaison se fera entre les librairies restantes.

Maintenant, nous allons analyser la qualité de l'extraction :

```
---- pypdf2 ----
File Decrypted (PyPDF2)
The Digital Millennium Copyright Act of 1998Copyright Office SummaryDecember 1998Page 9Each limit
ation entails a complete bar on monetary damages, and restricts theavailability of injunctive rel
ief in various respects.  (Section 512(j)).  Each limitationrelates to a separate and distinct fu
nction, and a determination of whether a serviceprovider qualifies for one of the limitations doe
s not bear upon a determination ofwhether the provider qualifies for any of the other three.  (Se
ction 512(n)).The failure of a service provider to qualify for any of the limitations in section5
12 does not necessarily make it liable for copyright infringement.  The copyrightowner must still
 demonstrate that the provider has infringed, and the provider may stillavail itself of any of th
e defenses, such as fair use, that are available to copyrightdefendants generally.  (Section 512(
l)).In addition to limiting the liability of service providers, Title II establishes aprocedure b
y which a copyright owner can obtain a subpoena from a federal courtordering a service provider t
o disclose the identity of a subscriber who is allegedlyengaging in infringing activities.  (Sect
ion 512(h)).Section 512 also contains a provision to ensure that service providers are notplaced
in the position of choosing between limitations on liability on the one hand andpreserving the pr
ivacy of their subscribers, on the other.  Subsection (m) explicitlystates that nothing in sectio
n 512 requires a service provider to monitor its service oraccess material in violation of law (s
uch as the Electronic Communications Privacy Act)in order to be eligible for any of the liability
 limitations.Eligibility for Limitations GenerallyA party seeking the benefit of the limitations
on liability in Title II must qualifyas a fiservice provider.fl  For purposes of the first limitati
on, relating to transitorycommunications, fiservice providerfl is defined in section 512(k)(1)(A) a
s flan entityoffering the transmission, routing, or providing of connections for digital onlinecom
munications, between or among points specified by a user, of material of the user™schoosing, with
out modification to the content of the material as sent or received.fl  Forpurposes of the other t
hree limitations, fiservice providerfl is more broadly defined insec 512(k)(l)(B) as fla provide
r of online services or network access, or the operatorof facilities therefor.flIn addition, to be
 eligible for any of the limitations, a service provider mustmeet two overall conditions:  (1) it
 must adopt and reasonably implement a policy ofterminating in appropriate circumstances the acco
unts of subscribers who are r peatinfringers; and (2) it must accommodate and not interfere with
fistandard technicalmeasur s.fl  (Section 512(i)).  flStandard technical measuresfl are defined as me
asuresthat copyright owners use to identify or protect copyrighted works, that have beendeveloped
 pursuant to a broad consensus of copyright owners and service providersin an open, fair and volu
ntary multi-industry process, are available to anyone on
```

Figure 7: Pypdf2, résultat de l'extraction du texte

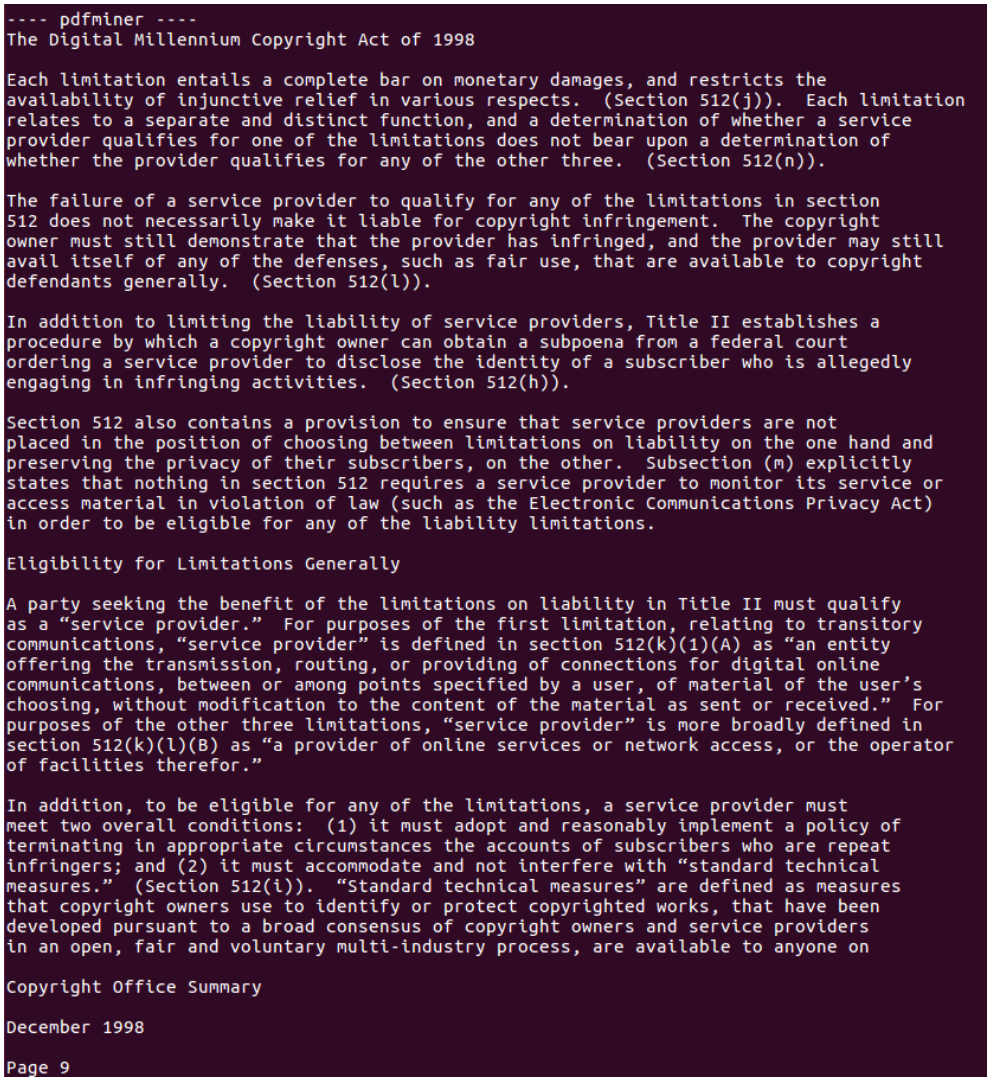

Figure 8: Pdfminer3k, résultat de l'extraction du texte

Dans ce dernier test, Pdfminer3k semble meilleur car il réussit à mieux décoder certains caractères spéciaux. Pour rappel ce test, a été réalisé sur un petit échantillon de documents.

Recherche texte

La prochaine étape va cette fois-ci permettre de localiser du texte dans le document.

Ce point devient déjà très important pour notre analyse. En effet, lors d'une recherche de texte, notre navigateur doit mettre en évidence la zone où le mot recherché a été trouvé, par conséquent notre librairie doit être capable de retourner les coordonnées de la zone (bounding box) dans laquelle la correspondance a été trouvée.

La suite des tests va seulement être faite avec Pdfminer3k car les autres librairies trouvées pour ce type de tâche ne font qu'envelopper Pdfminer.

Lors de l'exploration de notre document, Pdfminer3k va retourner un arbre (voir figure 8) avec les objets qu'il a reconnus.

Ensuite, nous aurons qu'à naviguer dans ces objets jusqu' au niveau « LTTextLine » dans lequel nous aurons l'information de son contenu. De ce fait, nous pourrons le comparer avec le mot recherché et dans le cas correspondant, retourner les coordonnées de son emplacement dans le fichier.

Ces coordonnées appelées « bounding box » correspondent aux quatre extrémités de notre objet. De cette façon nous saurons exactement où notre élément se trouve dans notre document.

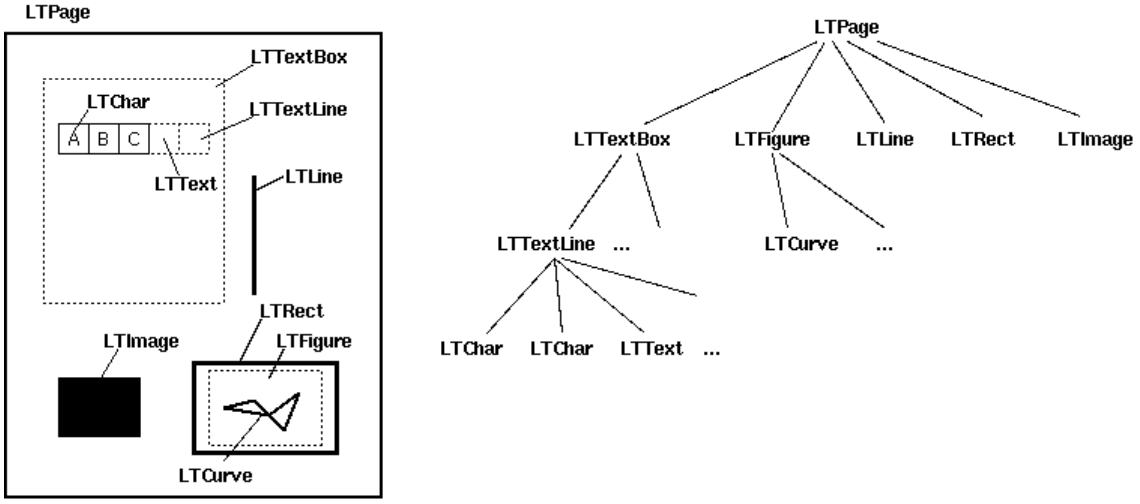

Figure 9: Structure layout Pdfminer (Levia3, 2017)

Dans ce test, un PDF de 170 pages a été utilisé, le script est disponible à l'annexe VII.

Le résultat est le suivant :

```
(OCR) christian@ubuntu-lenovo:~/virtenvs/OCR$ python script.py
---- pdfminer  ----
WARNING:root:Cannot locate objid=3852
Results:
[[[380.31722266200006, 181.471998, 452.18833710000007, 194.5520089], [281.90179451000006, 534.758998,
 353.77290894800007, 547.8390089], [298.7625067424, 468.59899800000005, 368.8016123809999, 481.679008
90000004], [120.66466956, 113.071998, 192.53578399799997, 126.1520089], [119.132, 325.758998, 185.993
9175364, 339.0244636], [359.181, 309.499998, 427.79182081199997, 322.7654636], [148.8404857671, 672.2
33998, 218.66820577500002, 685.3140089], [328.4104575818, 710.2089980000001, 399.1603129036001, 723.2
890089], [419.86238070939993, 673.822998, 490.8174907476998, 686.9030088999999], [393.11616933000005,
 202.311998, 464.98728376800005, 215.39200889999998], [370.478, 439.571998, 439.088820812, 452.837463
6], [171.14082948200002, 517.772998, 243.01194392000005, 530.8530089], [273.919, 657.5639980000001, 3
42.52982081199997, 670.8294636], [309.01617197400003, 114.391998, 380.88728641200004, 127.4720089], [
302.216529944, 695.0019980000001, 371.344005732, 708.0820089], [325.369758006, 602.1609980000001, 394
.4223973679999, 615.2410089], [324.4964030560001, 715.973998, 393.54904241800006, 729.0540089], [362.
39287100000007, 472.088998, 432.8547479, 485.1690089]], [9, 12, 12, 15, 28, 33, 49, 51, 67, 67, 72, 7
8, 110, 113, 121, 148, 149, 157]]
Elapsed time: 8.160603
```

Figure 10: Résultat d'une recherche pour la localisation d'un mot

Dans un tableau à dimension, nous retrouvons les coordonnées du bounding box et dans l'autre dimension, la page dans laquelle nous avons trouvé la correspondance. Pour notre besoin actuel, ce résultat nous convient parfaitement.

Table des matières

Le dernier point pour valider notre librairie est l'extraction de la table des matières. Cette opération se fait très aisément avec Pdfminer3k.

Le script reste le même que dans le test précédant (annexe VII).

Analyse du résultat :

```
(OCR) christian@ubuntu-lenovo:~/virtenvs/OCR$ python script.py --mode=toc
---- pdfminer  ----
WARNING:root:Cannot locate objid=3852
1 Titlepage
1 Abstract
1 1 General introduction
2 1.1 Introduction
3 1.1.1 Plastids adapt to changes in environmental conditions
3 1.1.2 Chloroplasts - a primary site of diverse metabolic pathways
3 1.1.3 Regulatory roles of plastids in isoprenoid compounds biosynthesis
3 1.1.4 Prenylquinone biosynthesis also derives from the MEP pathway
3 1.1.5 Tocochromanols essential plastid antioxidant metabolites
3 1.1.6 Plastoglobules are more than lipid droplets
2 1.2 General thesis outline
1 2 Unexpected roles of plastoglobules (plastid lipid droplets) in vitamin K1 and E metabolism
2 2.1 Introduction
2 2.2 Plastoglobules change with plant developmental stages and function as microdomains for (prenyl-) lipid metabolism
2 2.3 Vitamin K1 in the thylakoids and plastoglobules
2 2.4 Tocopherol cyclase (VTE1) located at plastoglobules fulfills diverse roles in the tocopherol metabolism
2 2.5 VTE1 and NDC1 enzymes are directly implicated in the redox cycle of -tocopherol
2 2.6 ABC1-like kinases affect prenyl lipid composition of the chloroplast
2 2.7 Conclusion
1 3 Lipid antioxidant and galactolipid remodeling under temperature stress in tomato plants
2 3.1 Introduction
2 3.2 Material and Methods
3 3.2.1 Plant material and stress treatments
3 3.2.2 Determination of photosynthetic parameters
3 3.2.3 Chlorophyll quantification
3 3.2.4 Lipid profiling
```

Figure 11: Résultat d'une extraction de la table des matières depuis un PDF

Même si l'extraction de la table des matières se passe très bien, cet outil à un gros défaut. Nous n'avons aucune information sur les pages de destination de ces éléments.

Ce problème vient du fait que la structure des PDF n'a pas été normalisée à ces débuts, causant des différences lors de leur création. C'est pour cette raison que les développeurs de Pdfminer3k n'ont pas intégré cette fonctionnalité dans leur outil.

3.5.3. Choix

Après cette analyse, nous constatons que dans l'environnement Python, l'extraction de texte avec les outils analysés n'offre pas de solution viable.

Vu ses défauts lors du dernier test, nous avons malheureusement dû aussi exclure Pdfminer de notre projet.

3.6. Problèmes rencontrés

Lors de cette analyse, nous constatons que pour l'extraction de texte, les outils que nous avons trouvé écrits en Python ne sont pas assez complets pour répondre aux besoins de notre navigateur mais dans la solution actuelle, Poppler est capable de résoudre ce type de problème. De plus, lors du test pour le rendu des fichiers PDF, nous avons constaté que cette librairie été déjà utilisée par pdftoppm.

Le problème à ce jour est que Swig, outil permettant l'interfaçage avec Poppler, devient incompatible lors d'une mise à jour de Poppler. Après un dernier essai de mise à niveau, la version maximale compatible de Poppler avec Swing était la version 0.49 (novembre 2016). Cela reste insatisfaisant pour l'équipe du RERO sachant que Poppler est actuellement à la version 0.67.

A ce moment de notre analyse, la meilleure solution est de trouver un nouvel outil capable de wrapper la librairie Poppler.

3.7. Wrapping Poppler

3.7.1. Etat de l'art

Dans ce dernier chapitre, nous allons analyser quel outil utiliser pour pouvoir interagir avec Poppler depuis notre code Python. En effet, Poppler est écrit en C/C++ et notre objectif est de pouvoir appeler des fonctions ou utiliser des objets Poppler en Python. La principale difficulté dans cette approche est la conversion des données mémoires de C/C++ à Python (retour des fonctions) ou Python vers C/C++ (arguments des fonctions).

De plus, l'utilisation de Poppler comporte des avantages. Poppler est utilisé par différentes applications de lecteur PDF dans le monde de l'open source ce qui nous garantit une bonne qualité du rendu et de l'extraction de texte. Dans les analyses précédentes, nous avons aussi constaté que certains fichiers PDF générés ne respectent pas complétement la norme PDF 1.7. Cette norme n'est pas si simple comme le montre le document de spécification qui comporte 756 pages. Il nous faut donc un outil robuste qui soit capable de naviguer dans la structure du PDF pour récupérer les informations du document. Le dernier avantage est que grâce à l'utilisation de Poppler, nous pouvons effectuer toutes les manipulations requises sans avoir besoin d'installer encore d'autres librairies et ainsi augmenter le nombre des dépendances.

Dans le marché, il existe différentes librairies capables de satisfaire notre besoin.

La liste des solutions retenues est la suivante :

- SIP
- Cython
- PyBindGen

SIP

SIP est un outil écrit en C/C++ et à son origine, il avait été développé pour la librairie PyQt mais par la suite, il a été utilisé dans d'autres domaines. Celui-ci est considéré comme une alternative à Swig.

Licence du projet GPLv2.

Cython

La syntaxe de Cython est très similaire à celle de Python mais elle supporte également du C/C++. Le premier intérêt de Cython était d'améliorer les performances de Python. Dans certains cas de figure, ce gain peut atteindre un facteur de 100.

Cython d'ailleurs automatise la conversion entre types Python et C, la gestion des erreurs et la compatibilité entre Python2 et Python3.

Licence Apache 2.0.

PyBindGen

PyBindGen est un autre outil permettant de créer des liaisons entre Python et C/C++. Il fournit également un outil capable d'analyser les en-têtes C et C++. Son problème est la compatibilité avec les versions de Python3. À ce jour, il supporte seulement les versions supérieures à la 3.3. De plus, l'outil d'analyse des fichiers en-tête n'est pas compatible avec Python3.

Licence LGPL v2.1.

Cython est sous licence Apache 2.0 qui n'est pas compatible avec notre projet sous licence GPLv2. Par contre, nous utilisons Cython que pour la génération du fichier pour le wrapping et ce fichier généré peut être utilisé dans notre projet.

SIP	Cython	PyBindGen
	Fichier de sortie compatible	

Tableau 6: Analyse wrapping, compatibilité des licences

3.7.2. Comparaison des solutions existantes

La comparaison sera faite selon les informations récoltées lors des recherches.

PyBindGen est le premier exclu car la compatibilité avec les versions de Python3 le rend instable pour notre implémentation.

Cython très proche de Python simplifie et unifie la syntaxe dans le serveur. De plus, lors des recherches, un projet pour l'extraction de texte utilisant la combinaison Cython-Poppler a été trouvé (github.com/izderadicka/pdfparser). Ce projet nous permettrait d'avoir une base pour notre interfaçage.

La communauté autour de Cython est aussi importante. A ce jour, le projet compte sur GitHub 244 contributeurs.

SIP est considéré comme l'alternative à Swig. Vu leur similitude, on pourrait avoir à nouveau des problèmes lors de l'interfaçage.

3.7.3. Choix

Suite à notre analyse, notre choix s'est porté sur Cython.

Les points déterminants ont été les suivants :

- Syntaxe très proche de Python
- Une base de projet implémentant le wrapping entre Cython-Poppler

L'unique modification que nous devons apporter à notre projet est sa licence. En effet, la base de projet trouvé sur Github est sous licence GPLv3 (Zderadicka, 2018). Pour rentrer dans la compatibilité, nous devons modifier la licence du projet et utiliser aussi la licence GPLv3, ce changement de licence n'impacte pas nos analyses précédentes.

3.7.4. Implémentation

Pour pouvoir implémenter notre wrapping, nous devons avant tout créer notre fichier au format pyx. Ce fichier sera divisé en deux parties.

La première partie correspond au lien entre Cython et Poppler. Ici, toutes les classes externes que nous voulons utiliser et leurs méthodes/constructeurs devront être déclarées.

Un exemple tiré de notre fichier :

```
cdef extern from "Catalog.h":
    cdef cppclass Catalog:
        Object *getOutline()
        int findPage(int num, int gen)
        int getNumPages()
        LinkDest *findDest(GooString *name)
```

Figure 12: Cython, déclaration header externe

Ci-dessus, nous faisons référence au fichier en-tête de la classe avec laquelle nous voulons interagir. Seules les classes et les méthodes que nous avons besoins seront implémentées.

Cela implique que nous devons connaitre exactement quelles fonctions de la librairie nous allons utiliser. Dans notre cas, une analyse de l'ancien serveur a permis de déterminer quelles méthodes étaient nécessaires pour implémenter notre solution.

Dans la deuxième partie du document, les éléments accessibles depuis Python seront déclarés. Ci-dessous, un exemple en image :

```
cdef class Flow:
    cdef:
        TextFlow *flow
        TextBlock *curr_block

    def __cinit__(self, Page pg):
        self.flow=pg.curr_flow
        self.curr_block=self.flow.getBlocks()

    def __iter__(self):
        return self

    def __next__(self):
        cdef Block b
        if not self.curr_block:
            raise StopIteration()
        b=Block(self)
        self.curr_block=self.curr_block.getNext()
        return b
```

Figure 13: Cython, implémentation de la classe accessible depuis Python

Pour terminer, nous devons simplement configurer le fichier setup.py dans le projet pour que le module de wrappage soit généré lors de l'installation du serveur.

3.7.5. Fonctionnalités implémentées

La librairie trouvée fournit déjà la possibilité d'extraire du texte depuis une source PDF. Les fonctionnalités non importantes telles que la taille du texte et sa couleur n'ont pas été prises en compte dans notre fichier.

En revanche, nous avons enrichi cette librairie en y ajoutant :

- L'extraction de la table des matières (annexe VIII)
- L'extraction des métadonnées du document (annexe IX)
- Le rendu vers une image (annexe X)
- Fonctions pour la récupération d'informations sur le document (taille, échelle, …)

Ces fonctionnalités n'ont pas été simples à implémenter car nous avons été confrontés à des problèmes de types différents et à manipuler des pointeurs.

3.7.6. Difficultés rencontrées

Cython prend automatiquement en compte les conversions de types simples (int, float, …) mais la difficulté surgit lorsque qu'on travaille avec des types non gérés par Cython. De plus, les développeurs de Poppler ont créé leurs propres types (GBool, Guint, …).

```
# Types Definition
ctypedef bool GBool
ctypedef unsigned char Guchar
ctypedef unsigned short Gushort
ctypedef unsigned int Guint
ctypedef unsigned long Gulong
ctypedef long long Goffset
ctypedef Guchar SplashColor[4]
ctypedef Guchar *SplashColorPtr
ctypedef unsigned int Unicode
```

Figure 14: Définition des types Poppler dans le fichier de wrapping

Lors l'extraction de la table des matières, nous avons dû utiliser des fonctions spécifiques comme PyUnicode_FromUnicode, PyInt_FromLong et d'autres pour pouvoir récupérer des types C en Python.

```
self.newOutlineLevel(&curr, <Catalog*>catalog, childs, level+1)
        PyDict_SetItemString(local_dic, "label", (PyUnicode_FromUnicode(<Py_UNICODE
                             *>title, titleLen)))
        if (page_number < 1 or page_number > catalog.getNumPages()):
          PyDict_SetItemString(local_dic, "page_number", None)
        else:
          PyDict_SetItemString(local_dic, "page_number",
                               PyInt_FromLong(page_number))
        if (PyList_Size(childs) > 0):
          PyDict_SetItemString(local_dic, "childs", childs)
        PyList_Append(myList, local_dic)
```

Figure 15: Extrait du code pour l'extraction de la table des matières

Lors du wrapping, une autre difficulté est la manipulation des pointeurs. Dans Poppler, nous nous retrouvons souvent à manipuler des pointeurs ce qui implique que lorsque nous les récupérons, nous devons connaitre leur taille afin de récupérer les informations correctement.

Par exemple, dans la classe « Image » de notre wrapping, nous renvoyons l'image du PDF grâce à la fonction « getBitmap ».

```
cdef class Image:
  cdef:
      SplashOutputDev *splash
      SplashBitmap *bitmap
      double scale
      Document pdf
      bytes data

  def __cinit__(self, int page, Document pdf, int max_width, int max_height):
      self.pdf = pdf
      self.splash = new SplashOutputDev(splashModeRGB8, 3, False, [255,255,255], True,
                                        splashThinLineDefault, False)
      self.splash.setFontAntialias(True)
      self.splash.setVectorAntialias(True)
      self.splash.startDoc(self.pdf.doc)
      self.scale = self.getOptimalScale(max_width, max_height, page)
      self.pdf.doc.displayPage(<OutputDev*>self.splash, page, 72*self.scale, 72*self.scale, 0,
                               True, True, False)
      self.bitmap = self.splash.getBitmap()

  def getScale(self):
      return self.scale

  def getBitmap(self):
       data =
       <bytes> (<char*>self.bitmap.getDataPtr())[:3*self.getWidth()*self.getHeight()]
      return data
```

Figure 16: Extrait du code de la classe Image (wrapping)

Le premier piège dans cette fonction était la taille de notre pointeur car il fallait tenir en compte que notre image avec le mode RGB8 comporte trois bytes par pixel.

Le deuxième problème que nous avons rencontré est la déclaration du constructeur « SplashOutputDev ». À la base, le constructeur contient des paramètres par défaut mais lorsqu'on utilise Cython, ces paramètres ne sont pas pris en compte correctement et cela cause des problèmes d'interprétation et l'image n'était pas extraite conformément.

4. Frontend

4.1. Framework JavaScript

Pour le choix du framework JavaScript, l'analyse sera effectuée sur les trois grands framework qui se concurrencent en ce moment : Angular 6, React et VueJS.

4.1.1. Etat de l'art

Angular 6

Développé et maintenu par Google, ce framework frontend JavaScript se base sur du TypeScript et il n'a rien à voir avec la version AngularJS car il est orienté "composants". Ce projet open-source sous licence MIT est idéal pour des sites web SPA (single page application). Étant multi-plateforme, ce framework permet le développement pour différents types d'appareils (smartphones, tablettes, …).

Les créateurs d'Angular ont rendu cette dernière version encore plus rapide, plus facile d'utilisation et moins lourde que les versions précédentes. La version 6 d'Angular a également simplifiée l'intégration de ces mêmes composants dans les pages web existantes développées en JavaScript/JQuery.

Ce framework prêt à l'utilisation, incorpore déjà toutes les fonctions essentielles pour le développement d'une application.

En image, la structure de son fonctionnement.

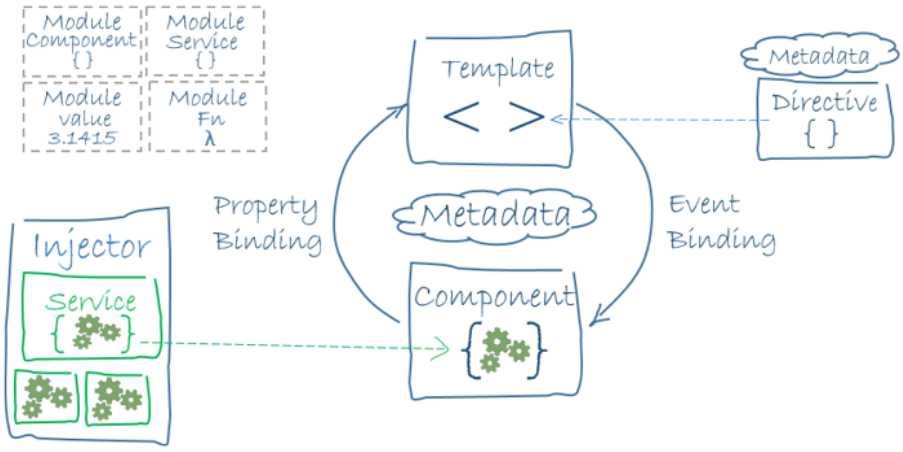

Figure 17: Diagramme de fonctionnement Angular(Angular, 2018)

Les points forts d'Angular sont :

- Chargement possible dans un WebWorker
- Ajout d'un utilitaire cli : @angular/cli
- Server rendering (génération du contenu depuis le serveur)
- Migration depuis une ancienne version automatisée
- Proche des langages OOP (Python,Java, C#)
- Bonne documentation

Les points faibles :

- Courbe d'apprentissage lente pour nouvelle équipe de développeurs

React 16.3.2

En premier lieu, il faut préciser que React n'est pas un framework mais bien une librairie open source sous licence MIT. Le but principal de React est la création de sites SPA. Cette librairie est écrite en JavaScript avec l'extension JSX.

Comme précisé ci-dessus, React n'est pas un framework et cela peut être un point en même temps positif et négatif.

L'intégration de nouveaux outils sera gérée par le développeur. Il devra également choisir et tester la meilleure solution dans le gestionnaire de packages (npm, yarn, …). Cela comporte une meilleure maîtrise dans le choix des versions des libraires afin de ne pas créer des conflits. D'un autre côté, cela permet d'utiliser seulement ce dont nous avons besoin et ainsi alléger notre l'application.

Il existe aussi une version native, React Native, qui permet de faire des applications pour les appareils mobiles ou pour les tablettes.

React peut être combiné avec plein d'autres librairies selon des besoins spécifiques. React utilise du JSX ES6/7 à la base mais on peut le configurer pour utiliser du TypeScript.

En image, le schéma de son fonctionnement :

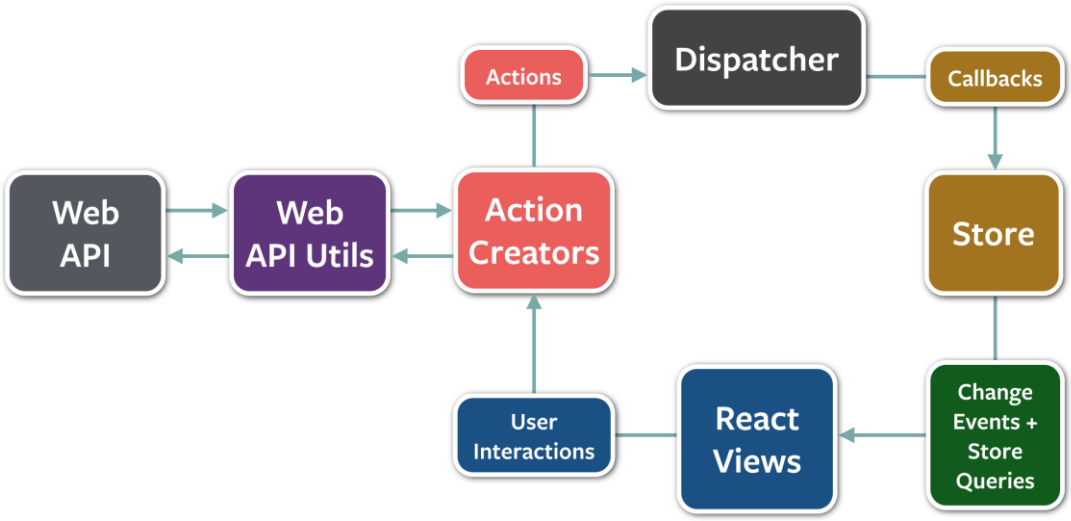

Figure 18: Architecture de Flux React (GitBooks, s.d.)

Points positifs :

- Virtual DOM car il permet de mettre à jour uniquement les composants touchés par la modification
- Communauté grandissante dans le web et dans le milieu professionnel
- Migrations aisés grâce aux outils fournit par Facebook.
- Ne requiert pas de nouvelles connaissances. HTML et JavaScript suffisent.
- Flux des donnés unidirectionnel avec les « props »
- Flux de donnés bidirectionnel avec les « states »

Points faibles :

- Le fait de ne pas être un framework laisse beaucoup de choix aux développeurs qui peuvent perdre du temps à la recherche des bons outils (unit test, …)
- Utilise beaucoup de RAM pour stocker la Virtual DOM

VueJS

Pour terminer, une petite description du dernier arrivé sur le marché, VueJS. Si les deux précédents sont des produits nés de grandes entreprises telles que Facebook ou Google, celui-ci est né d'un groupe de développeurs JavaScript. Parmi eux, Evan You fondateur de VueJS qui a travaillé

chez Google sur des nombreux projets avec AngularJS. Après avoir quitté Google, il a voulu reprendre ce qu'il trouvait intéressant chez AngularJS pour créer un outil plus léger et plus performant.

Multi-plateforme, VueJS est un framework écrit en JavaScript sous licence MIT.

Autre que pour des applications SPA simples, il peut être adopté pour des projets plus complexes. Son intégration dans une page web JavaScript se fait aussi aisément.

Dans ces projets, VueJS permet d'utiliser le JSX mais aussi le TypeScript.

VueJS est considéré comme un milieu entre React et Angular.

Points forts :

- Sa petite taille (environ 20 KB) lui permet d'atteindre de bonnes performances
- La réutilisation des templates permet de gagner du temps lors du développement
- Utilisation de VueJS pour des applications SPA mais aussi pour des applications plus complexes
- Intégration facile dans une page web JavaScript, ceci n'aura pas de répercussion sur les performances du système
- Courbe d'apprentissage très rapide
- Documentation officielle bien détaillée.

Points faibles :

- Marché encore très petit. Vue son jeune âge, la communauté qui l'entoure est encore faible
- Barrière de la langue. Né grâce à des fonds chinois, au commencement beaucoup de documentation était en chinois. Cela a rendu son intégration dans le marché plus difficile

En image, le fonctionnement de VueJS :

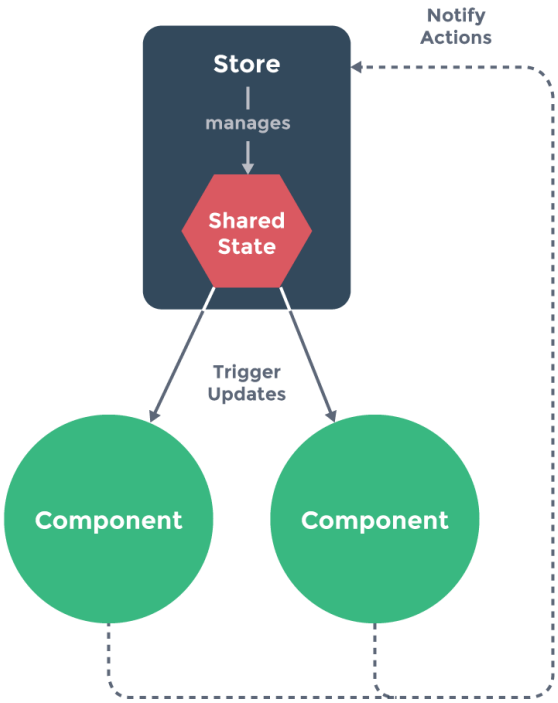

Figure 19: Architecture de VueJS (VueJS, s.d.)

Angular 6	React 16.3.2	VueJS
✓	✓	✓

Tableau 7: Analyse frameworks JS, compatibilité des licences

4.1.2. Comparaisons des solutions existantes

Maintenant, nous allons faire une comparaison sur différents critères afin que nous puissions trancher sur la technologie à utiliser pour le frontend.

Premier point de comparaison est la communauté qui tourne autour de ces technologies.

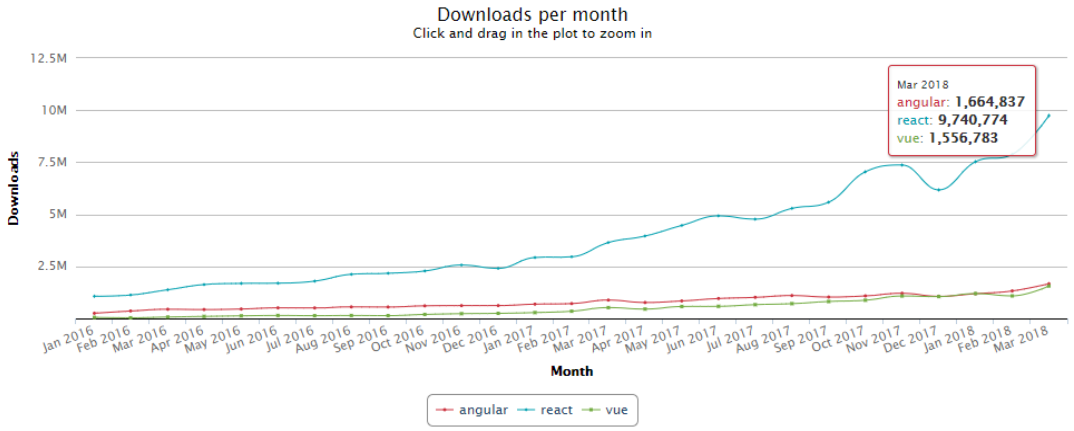

Figure 20: Nombre de téléchargements sur npm (Vorbach, s.d.)

Figure 21: Tendances des recherches sur Google

Nous constatons que React est largement le plus téléchargé sur npm. De ce fait, cela facilitera la recherche lors d'éventuel problème.

Points comparaison	Angular 6	React	Vue.js
Stable	Oui	Oui	Oui
Bonne documentation	Oui	Oui	Oui
Taille	130 KB	46 KB	20 KB
Code réutilisable	Oui	Seulement CSS	HTML + CSS
Basé composants	Oui	Oui	Oui
Vitesse apprentissage	Lente	Normale	Rapide
Communauté	Oui	Oui	Pas encore

Tableau 8: Tableau comparatif framework JS

4.2. Intégration dans un projet existant

Maintenant, nous allons voir des exemples concrets d'intégration de ces outils. Ce point sera déterminant car dans le but de notre travail, nous devons intégrer notre solution à la page web existante du RERO.

Intégration avec Angular 6

Un des points principaux que l'équipe d'Angular à voulu simplifier est l'intégration de composants Angular dans une application JavaScript existante. Ajouter un composant Angular jusqu'à la version 4 été faisable mais aujourd'hui cela se fait plus aisément.

La création du projet restera invariée. Nous devrons par la suite ajouter la fonctionnalité pour la création des « Angular Elements ». Ceci se fait facilement grâce à la console Angular.

```
ng add @angular/elements
```

Figure 22: Commande d'ajout pour la fonctionnalité "Angular Elements"

Ensuite, nous pourrons créer nos composants comme dans un projet standard.

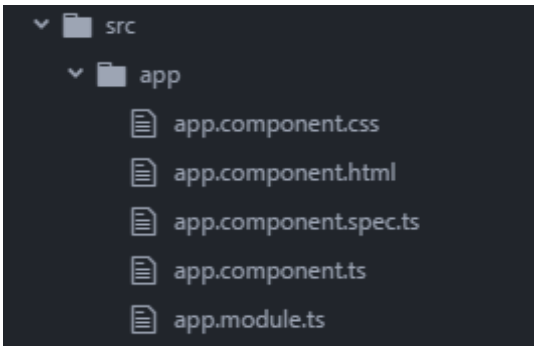

Figure 23: Structure d'un composant Angular

La structure des composants dans une application Angular est très intéressante car elle permet de séparer toutes les différentes couches et l'intégration des tests unitaires.

Le fichier clé pour pouvoir exporter notre composant est le suivant :

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule, Injector } from '@angular/core';
import { createCustomElement } from '@angular/elements';

import { AppComponent } from './app.component';

@NgModule({
  declarations: [AppComponent],
  imports: [BrowserModule],
  entryComponents: [AppComponent]
})
export class AppModule {
  constructor(private injector: Injector) {
    const hello = createCustomElement(AppComponent, { injector });
    customElements.define('app-root', hello);
  }
  ngDoBootstrap() {}
}
```

Figure 24: Fichier app.module.ts

Nous devons modifier le constructeur pour informer le framework de quelle manière assembler notre application.

Reste une dernière chose à modifier dans notre fichier **package.json** :

```
"build": "ng build --prod --output-hashing=none"
```

Figure 25: Modification du fichier package.json

Après avoir lancé la commande build avec npm, nous obtenons dans le dossier « dist » les éléments que nous nécessitons pour implémenter notre composant dans la page web existante.

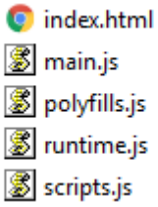

Figure 26: Fichiers générés par Angular

A ce stade, nous pouvons ajouter le composant dans une page web existante et inclure les fichiers JavaScript générés. Une démonstration est disponible à l'adresse suivante : https://codepen.io/Christian_Pereira/pen/qYzwqz.

Intégration avec React

Avec React, nous pouvons facilement ajouter les références des scripts dans l'actuel site comme tout autre script JavaScript.

```
<html>
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>React Integration</title>
    <script
src="https://cdnjs.cloudflare.com/ajax/libs/react/15.4.2/react.js"></script>
    <script src="https://cdnjs.cloudflare.com/ajax/libs/react/15.4.2/react-
dom.js"></script>
    <script src="https://cdnjs.cloudflare.com/ajax/libs/babel-
standalone/6.21.1/babel.min.js"></script>
  </head>
  <body>
  <section class="section">
    <div class="container">
      <h1 class="title">React intégration</h1>
      <p class="subtitle">Legacy website!</p>
      <div id="root"></div>
    </div>
  </section>
  </body>
</html>
```

Figure 27: Intégration React

Dans cet exemple, nous avons également ajouté la référence de la librairie Babel pour pouvoir utiliser ES6 avec React.

Ensuite, nous pouvons facilement écrire le composant dans la page web.

Une démonstration de cette implémentation est disponible à l'adresse suivante : https://codepen.io/Christian_Pereira/pen/Pergwz

Intégration avec VueJS

Comme pour React, nous pouvons facilement ajouter VueJS au projet existant grâce au CDN.

```
<html>
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>VueJS Integration</title>
    <script
src="https://cdn.jsdelivr.net/npm/vue@2.5.13/dist/vue.js"></script>
  </head>
  <body>
  <section class="section">
    <div class="container">
      <h1 class="title">VueJS intégration</h1>
      <p class="subtitle">Legacy website!</p>
      <div id="root">{{name}}</div>
    </div>
  </section>
  </body>
</html>
```

Figure 28: Intégration VueJS

Dès maintenant, nous pouvons démarrer notre composant en VueJs.

Ici, la démonstration https://codepen.io/Christian_Pereira/pen/pVXxmj

4.2.1. Choix

Nous constatons que React et VueJS ont une intégration différente d'Angular. Le point positif en faveur d'Angular est que les composants sont plus facilement réutilisables. Si notre composant n'est pas complexe, React et VueJS peuvent très bien faire l'affaire mais dès que sa complexité augmente cela peut vite devenir problématique si nous n'implémentons pas une bonne architecture.

Le choix va également découler de pleins de facteurs. Chaque technologie a ses points forts et ses points faibles mais la décision finale va dépendre du développeur et de ses habitudes.

Dans le cadre de ce projet, le choix va se porter sur Angular6 car son framework incorpore déjà tous les outils qui permettent d'écrire un code propre (tslint, tests unitaires, …) et ainsi nous faire gagner du temps lors du développement de notre prototype. De plus avec Angular, nous pouvons développer notre application dans notre environnement de programmation sans avoir besoin d'ajouter des scripts dans la solution existante qui pourrait vite devenir lourde à maintenir.

4.3. Framework HTML/CSS

4.3.1. Etat de l'art

Aujourd'hui, ils existent différents frameworks CSS de bonne qualité. Le but de cette analyse est de trouver un framework facilement utilisable et si possible permettant de gagner du temps lors du développement.

Dans l'état de l'art, nous aurons des frameworks CSS expressément liées au framework JavaScript et certains indépendants. Pour rappel, notre choix pour le framework JavaScript s'est porté sur Angular 6.

Bootstrap

Bootstrap est sûrement le framework le plus populaire de ces dernières années. Il a une des plus grandes communautés derrière lui.

Sous licence MIT, il fait partie d'un projet né de Twitter. En ce moment, la dernière version disponible est la 4.1.1. Bootstrap bénéficie d'une expérience de 7 ans et peut être utilisé par n'importe quel framework analysé dans le chapitre précédent. Il existe aussi des versions spécifiques selon le framework choisi. Bootstrap utilisé du JQuery pours ses animations.

SemanticUI

SemanticUI est un autre framework CSS très connu. Il a une intégration dédiée à Angular. De plus, il est open source et sous licence MIT. Ce framework est encore en phase de développement mais la plupart de ces composants sont déjà disponibles.

Bulma

Bulma est un des derniers framework CSS arrivé sur le marché. Projet open-source sous licence MIT, il se développe très rapidement et s'adapte facilement à n'importe quel framework car il n'utilise pas de JavaScript. Bulma utilise le Flexbox et grâce au SASS, il peut être facilement customisé.

Bulma est aussi modulaire. En effet, nous pouvons importer seulement les fichiers SASS qui nous intéressent sans avoir besoin d'importer l'intégralité de la librairie. Ainsi, notre application reste plus légère.

Ant Design NG Zorro

Ce framework d'origine asiatique est né du projet Ant Design pour React. Il est sans aucun doute le dernier arrivé. Cette librairie permet d'intégrer facilement des composants déjà prêts à l'utilisation pour avancer plus rapidement dans le développement. Sa documentation est bien détaillée avec des exemples concrets. Licence de la librairie MIT.

Bootstrap	Semantic UI	Bulma	NG Zorro
✓	✓	✓	✓

Tableau 9: Analyse framework CSS, compatibilité des licences

4.3.2. Comparaison des solutions existantes

Bootstrap

Points positifs :

- Grande communauté
- Expérience de 7 ans
- Bonne compatibilité avec IE11

Points négatifs :

- Animations JQuery
- Customisation difficile

Bulma

Points positifs :

- Pas de Javascript/JQuery
- Equipe expérimentée

Points négatifs :

- Compatibilité de 90% avec IE11
- Intégration facile dans un projet

Semantic UI

Points positifs :

- Facile d'utilisation
- Bonne documentation

Points négatifs :

- Petite communauté
- Intégration Angular pas très active

Ant Zorro

Points positifs :

- Communauté active grandissante
- Intégration facile
- Composants prêts à l'utilisation

Points négatifs :

- Framework encore très jeune

Les quatre frameworks sont des bonnes solutions.

Bootstrap a une grande communauté derrière lui et sept ans d'expérience. Il a également une bonne compatibilité avec les navigateurs IE11. En revanche, sa customisation est moins facile et il contient du JQuery pour ses animations.

Bulma contient seulement du CSS ce qui le rend compatible avec tous les développements web. De plus, son utilisation est très simple et on peut facilement le customiser pour créer un meilleur style pour notre l'application. Son point faible est sa compatibilité avec IE11 où il est compatible qu'à 90%.

SemanticUI ne semble pas viser beaucoup sur son intégration avec Angular. Depuis quatre ans, il compte seulement 8 contributeurs et 119 commits sur son projet GitHub. En comparaison, Ant NG Zorro compte 39 contributeurs et 438 commit (39 releases) après six mois.

Pour terminer, Ant NG Zorro. Étant le dernier arrivé, il est le moins expérimenté mais il se base sur un projet qui a eu du succès parmi les développeurs React. Ces derniers mois, la communauté autour de cette technologie s'agrandit très rapidement. L'intégration de NG Zorro se fait aisément et ses composants permettent aux développeurs d'avancer plus rapidement.

4.3.3. Choix

Le choix pour cette analyse va se porter sur Ant NG Zorro. Même s'il est très récent, sa librairie et ses composants sont bien structurés et leurs intégrations se fait vraiment très aisément.

Grâce à ce type de solution, nous pourrons rapidement développer un prototype pour notre projet.

5. Description de l'application finale

Dans cette section, on va vous détailler la solution que nous avons proposée au RERO et ainsi que son fonctionnement.

5.1. Périmètre solution

Pour rappel, le périmètre de notre application est le suivant :

- Concevoir un serveur en Python sous forme de module Invenio (imposé par le RERO)
- Créer un wrapping avec Cython pour pouvoir utiliser Poppler (C++) depuis notre serveur
- Créer une interface web Angular pour pouvoir visualiser les documents

5.2. Périmètre tests

5.2.1. Backend

Le backend a été conçu pour être le plus stable possible. Pour cela, des tests unitaires ont été mis en place dès le début. Toutes les fonctionnalités ainsi que toutes les requêtes que notre serveur est capable de servir sont testées.

A ce jour, nous avons un taux de 97% de couverture du code. Le 3% restant correspond à des cas spécifiques (exceptions, erreurs, …) qui ne sont pas encore implémentés dans nos tests.

Le résultat des tests unitaires :

```
---------- coverage: platform linux, python 3.6.5-final-0 -----------
Name                                    Stmts   Miss  Cover   Missing
---------------------------------------------------------------------
invenio_multivio/__init__.py                8      0   100%
invenio_multivio/config.py                  7      0   100%
invenio_multivio/ext.py                    28      1    96%   67
invenio_multivio/image/__init__.py          1      0   100%
invenio_multivio/image/api.py              34      0   100%
invenio_multivio/image/views.py            42      0   100%
invenio_multivio/json/__init__.py           1      0   100%
invenio_multivio/json/api.py               48      0   100%
invenio_multivio/json/views.py             24      0   100%
invenio_multivio/pdf/__init__.py            1      0   100%
invenio_multivio/pdf/api.py               171     14    92%   139-144, 147-148, 221, 228, 230, 251, 270-271
invenio_multivio/pdf/views.py             103      0   100%
invenio_multivio/utils.py                   9      0   100%
invenio_multivio/version.py                 3      0   100%
invenio_multivio/views.py                   7      1    86%   41
invenio_multivio/xml/__init__.py            1      0   100%
invenio_multivio/xml/api.py                54      0   100%
invenio_multivio/xml/views.py              24      0   100%
---------------------------------------------------------------------
TOTAL                                     566     16    97%


========================== 104 passed in 2.93 seconds ==========================
```

Figure 29: Tests unitaires backend, couverture du code

Dans ces tests, nous vérifions aussi le fonctionnement de notre wrapping Cython sur Poppler.

5.2.2. Frontend

Côté frontend, les tests sont très basiques. Selon discussion avec l'équipe du RERO, nous allons simplement vérifier que nos composants se construisent correctement.

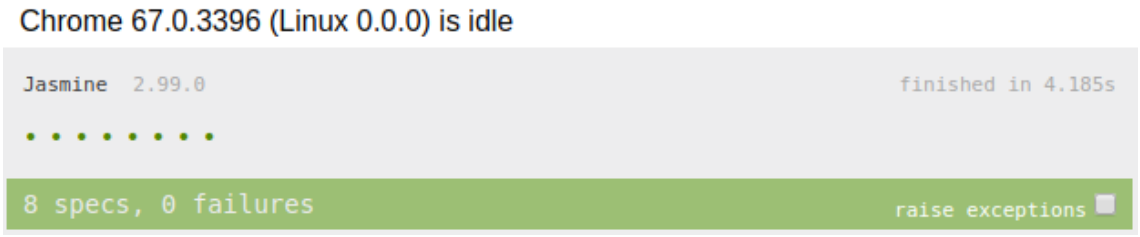

Figure 30: Tests unitaires frontend

La suite des tests est faite par un utilisateur (testeur) qui analysera l'application pendant son utilisation.

Le navigateur officiel pour nos tests est Google Chrome. En janvier 2018, il représentait plus de 58% des parts du marché.

5.3. Solution proposée

5.3.1. Backend

Comme nous l'avons déjà dit auparavant, le fichier XML décrivant notre publication est le point de départ de notre application. Selon les fonctionnalités demandées par le RERO, notre serveur sera également capable d'interpréter les fichiers JSON. La base du fonctionnement ne changera pas de la Figure 1.

Notre nouveau serveur est conçu sous forme d'API grâce à Flask et au Blueprints. Les Blueprints nous permettent d'avoir un serveur modulaire et ensuite avec Flask des routes spécifiques pour chacun de nos modules. Cela nous permet d'avoir différents points d'entrée selon le type de données avec lesquelles nous travaillons.

A ce jour, nous sommes capables de travailler avec : PDF, images, fichiers JSON et fichiers XML. Cela signifie que nous avons quatre points d'entrée principaux.

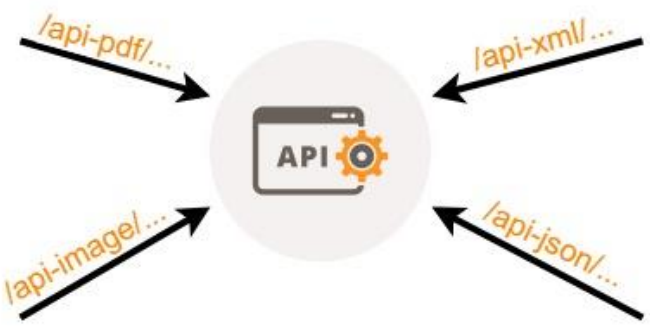

Figure 31: API, points d'entrée dans le serveur

Nous retrouvons aussi cette structure dans l'organisation du code.

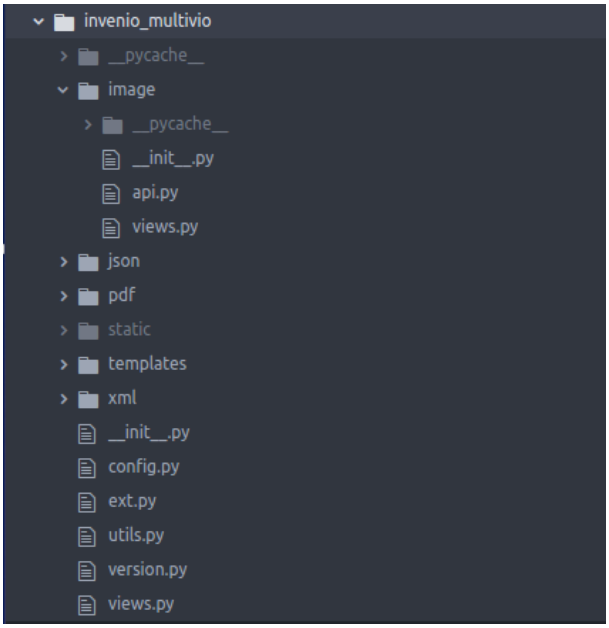

Figure 32: Organisation du code dans le serveur

Chaque dossier correspondant à un des points d'entrée est composé d'un fichier « api.py » et d'un fichier « views.py ».

Les points d'entrée sont définis dans le fichier « ext.py » grâce au Blueprints.

```
    def init_app(self, app):
        """Flask application initialization."""
        self.init_config(app)
        app.register_blueprint(views)
        app.register_blueprint(pdf_views)
        app.register_blueprint(json_views)
        app.register_blueprint(xml_views)
        app.register_blueprint(image_views)
        app.extensions['invenio-multivio'] = self
```

Figure 33: Fichier ext.py, ajout Blueprints

Le fichier « views.py » détermine les routes que nous acceptons dans ce point d'entrée.

Un exemple en image :

```
@views.route('/sizes/<path:path>/', methods=['GET'])
def get_sizes(path):
    """Retrive sizes image."""
    file_to_path = current_app.config.get('MULTIVIO_FILENAME_TO_PATH')
    path = file_to_path(path)
    if not path:
        abort(404)
    img = ImageProcessor(PIL_Image.open(path), path)
    sizes = img.get_sizes()
    return jsonify(sizes)
```

Figure 34: Exemple d'une route dans le fichier views.py

Le fichier « api.py » correspond à la classe que nous exposons pour notre API tandis que « views.py » ne fait qu'utiliser cette classe pour répondre aux requêtes reçues.

A ce stade du développement, notre serveur travaille avec des fichiers qui se trouvent dans un dossier « static », une des User Stories à suivre sera l'implémentation des URL.

Les réponses de notre serveur peuvent être de type image ou pour la plupart en format JSON.

Exemple de réponse :

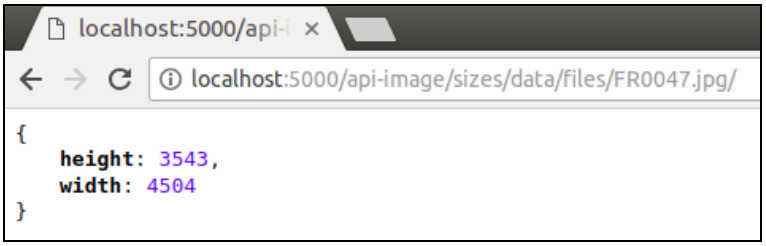

Figure 35: Exemple de réponse API

Pour terminer, le dossier « poppler » contient le script « mypoppler.pyx » (Cython) qui permet de générer notre module pour l'interfaçage avec la librairie Poppler (C++).

Le code source de ce projet est disponible à l'adresse :

https://github.com/ChristianPereiraHES/invenio-multivio

5.3.2. Frontend

Pour rappel, le choix de notre technologie frontend s'est porté sur Angular6.

Angular se base sur la programmation avec composants. Dans ces mêmes composants, nous avons la possibilité d'injecter des services. Le schéma suivant montre la structure actuelle de notre application.

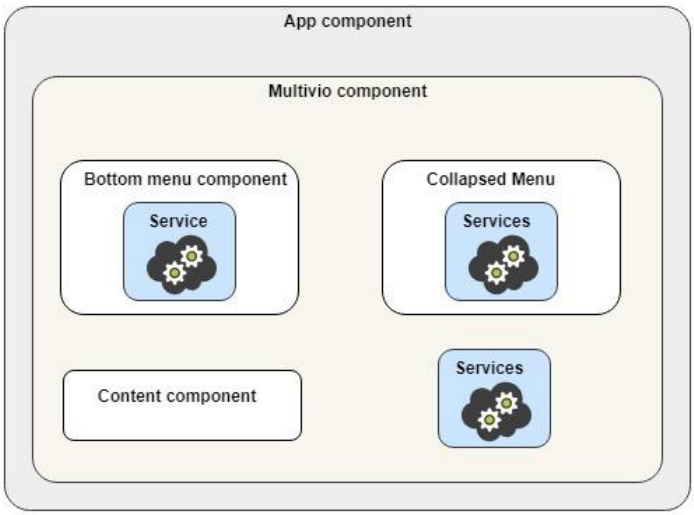

Figure 36: Frontend, structure des composants

Le composant Multivio est le cœur de notre application. À l'intérieur, on y retrouve le restant des composants ainsi que leurs services. Cette relation permet aux enfants d'émettre des messages vers le parent (Multivio). Celui-ci pourra ensuite, selon le message reçu, distribuer l'information ou apporter directement les modifications nécessaires.

Dans la solution proposée, nous avons à ce moment trois types de services. Le service « base » qui concerne toutes les requêtes communes pour tous types de document confondus. Au contraire, les services « document » et « image » correspondent aux requêtes spécifiques selon le type de fichier.

De cette façon, si un nouveau type de document venait s'ajouter, nous pourrions facilement l'implémenter et si nécessaire utiliser des autres composants spécifiques.

Comme énoncé pendant l'analyse des frameworks JavaScript, lors de leur génération les composants Angular comportent quatre fichiers :

- Un fichier pour le code CSS
- Un fichier pour le code HTML
- Un fichier pour les tests unitaires
- Un fichier pour la logique du composant

Cette structure permet de séparer le code de manière propre et d'aider sa maintenabilité.

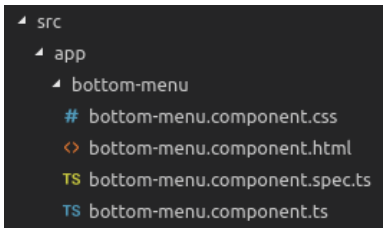

Figure 37: Structure d'un composant Angular

Notre navigateur comporte quatre composants :

- Multivio, layout et logique principale de notre navigateur
- Bottom-menu, le menu de navigation de notre navigateur
- Collapsed-menu, sous menu lors de la sélection d'un élément du menu principal
- Content, composant contenant l'image reçue depuis le server

Le code source de ce projet est disponible à l'adresse :

https://github.com/ChristianPereiraHES/invenio-multivio-ui

5.4. Fonctionnalités développées

Dans le dernier chapitre concernant la description de notre application, nous allons énoncer les principales fonctionnalités développées tout au long de notre travail.

5.4.1. Recherche de texte

Lorsque nous sommes dans un document de type PDF, nous pouvons réaliser une recherche de texte. Les 100 premières correspondances seront affichées. En sélectionnant un élément de la liste,

l'utilisateur sera directement redirigé vers la page où la correspondance a été trouvée. De plus, des zones colorées permettent de visualiser rapidement l'emplacement dans lequel les mots correspondants ont été trouvés.

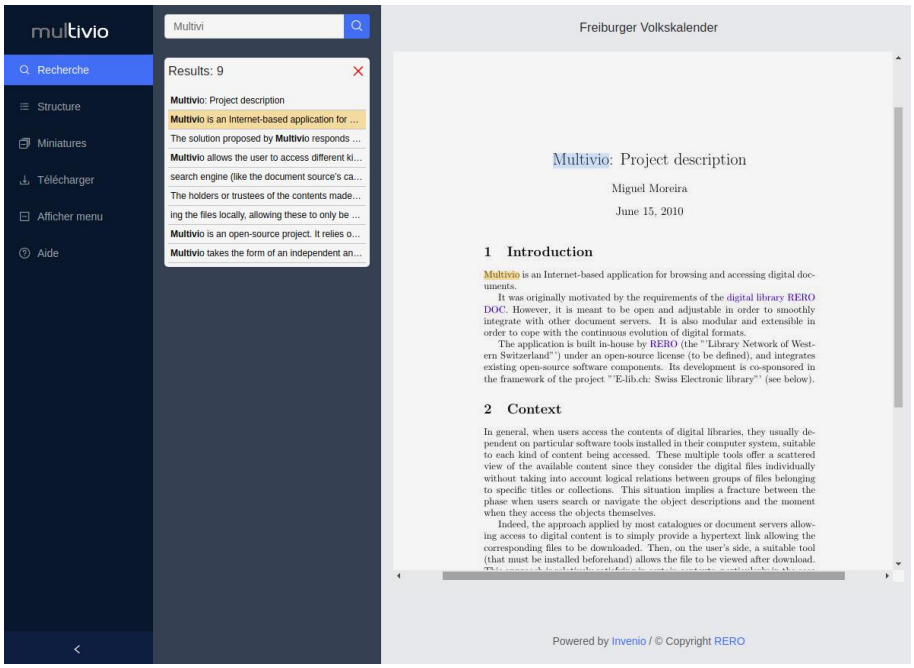

Figure 38: Recherche de texte dans Invenio Multivio

5.4.2. Affichage de la table des matières

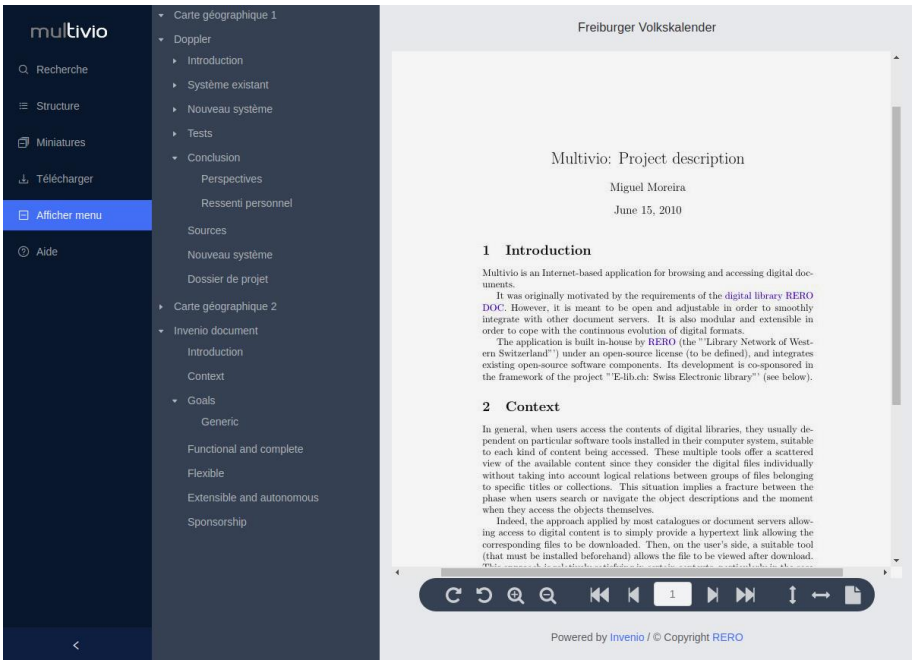

Figure 39: Table des matières, Invenio Multivio

Notre navigateur est capable de mixer des documents. Dans la Figure 39, par exemple, nous pouvons constater que notre table de matières comporte deux images et deux fichiers PDF.

Lors de la sélection d'un élément de la table des matières, la page du document sélectionnée sera affichée.

5.4.3. Affichages miniatures

L'application permet de visualiser les miniatures du document. Deux types d'affichage sont possibles, par liste ou en grille. Le chargement de ces mêmes miniatures se fait de manière asynchrone. Comme avec la table des matières, les miniatures peuvent servir de navigation.

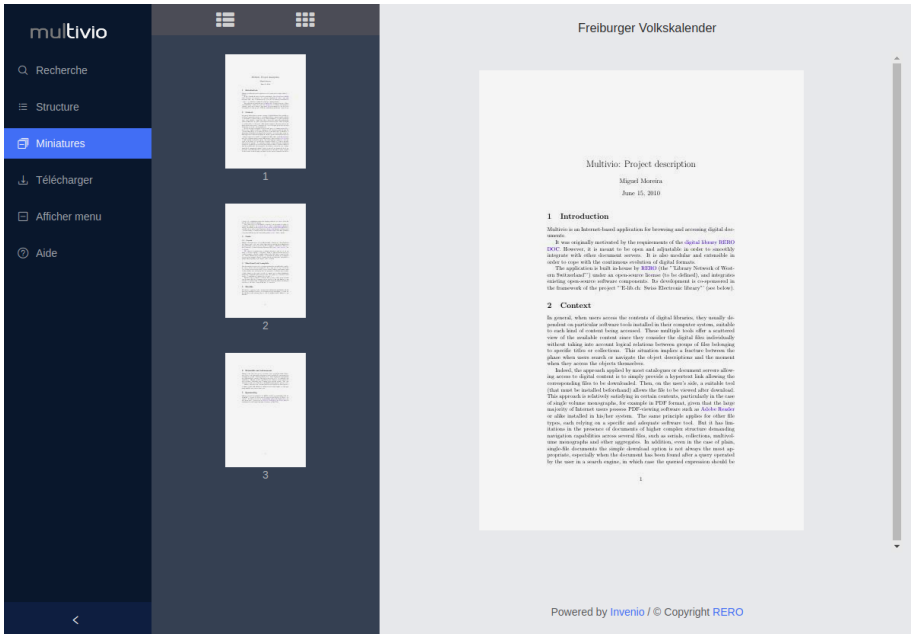

Figure 40: Affichage miniatures, Invenio Multivio

5.4.4. Navigation

Pour afficher le menu, nous avons deux possibilités. L'activer avec le menu de gauche (compatibilité avec les appareils tactiles) ou positionner la souris sur la zone concernée.

Grâce à ce menu, nous pouvons naviguer parmi les pages du document mais aussi modifier la taille de notre affichage.

Figure 41: Menu navigation, Invenio Multivio

5.4.5. Téléchargement fichier

Lorsque nous consultons un document, nous pouvons si nécessaire, le télécharger pour sauver une copie sur notre périphérique.

5.5. Améliorations possibles

Selon le Product Backlog élaboré en début du travail, nous avons encore quelques User Stories à implémenter.

Une fois ce travail effectué, notre application sera prête pour la mise en phase test en interne.

La compatibilité sur les différents systèmes et leurs navigateurs pourra être testée ainsi que l'expérience utilisateur. Un outil très intéressant pour ce type de test est le site web app.crossbrowsertesting.com. Ce site permet de lancer des machines virtuelles dans le navigateur en ayant aussi la possibilité de choisir le navigateur et sa version.

Une fois les tests réussis, une première version pourra être mise à disposition. Par la suite, on pourra l'enrichir en y ajoutant des fonctionnalités comme le support de fichier audio et vidéo. Ceci ne devrait pas poser problèmes car autant le serveur que le frontend ont été conçus pour être flexibles.

6. Gestion projet

Dans cette section, nous allons détailler de quelle manière nous avons géré le projet tout au long de sa durée.

6.1. Méthodologie

L'équipe du RERO travaillant avec la méthodologie Agile a établie au préalable le Product Backlog. Les tâches ont été prises selon la priorité établie par le RERO.

Johnny Mariéthoz, responsable du projet, a suivi le déroulement de ce travail tout au long de son développement et il a également validé les User Stories lors de nos séances.

Au total, six sprints ont été effectués. La durée des sprints a été établie à trois semaines mais selon l'avancement du projet et la disponibilité de parties prenantes, la taille des sprints a dû être adaptée.

En résumé, les sprints du projet :

Sprints	Durée	Story Points	Heures
Sprint 1	3 semaines	20	24
Sprint 2	3 semaines	31	46
Sprint 3	3 semaines	16	40
Sprint 4	2 semaines	16	26
Sprint 5	3 semaines	14	31
Sprint 6	4 semaines	48	80

Tableau 10: Résumés des sprints du projet

Les détails de la gestion du projet sont disponibles sur CD, fichier Product_Backlog.pdf

Total des heures des sprints accumulés : 247 heures.

En résumé, le but de chaque sprint en image :

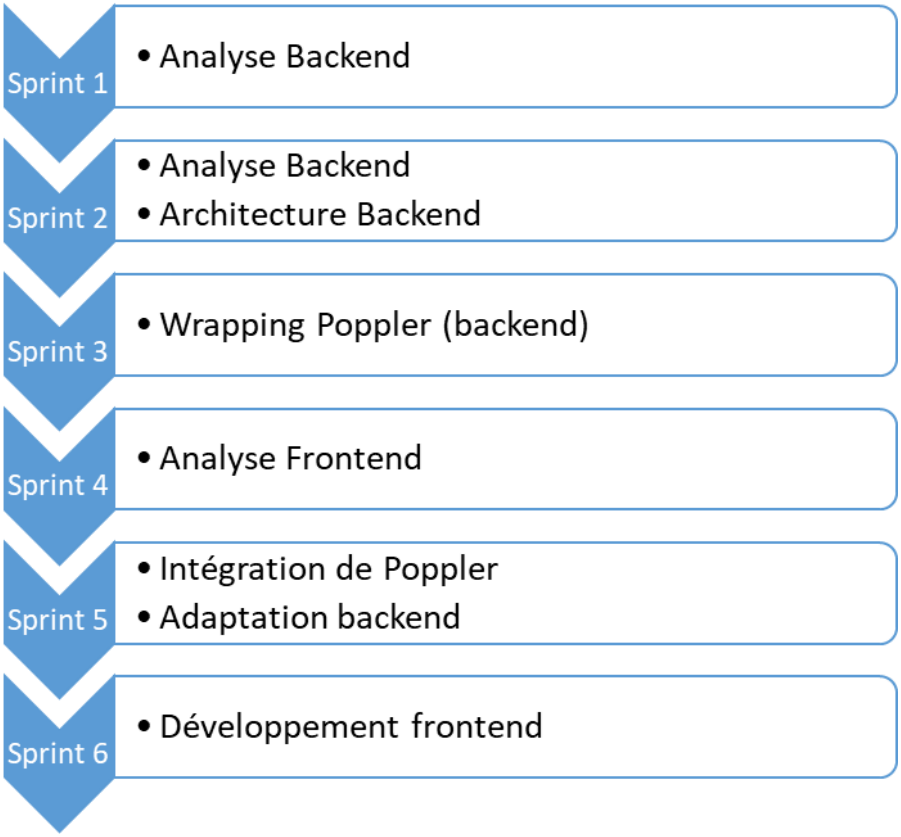

Figure 42: Déroulement du projet

Pendant l'évolution de l'application, des parties du rapport ont été écrites. Une fois le dernier sprint terminé, la rédaction du rapport final a été reprise.

L'écriture du rapport a pris environ 80 heures au total.

6.2. Outils

Pour faciliter la communication, l'outil Gitter a été utilisé.

Pour l'attribution des User Stories depuis le Product Backlog, la plateforme Taïga a été proposée par le RERO. Cependant, lors du déroulement du projet, nous avons passé à l'utilisation du fichier officiel de l'école pour la gestion des projets Agile. Même s'il reste un très bon outil pour la gestion de projets, Taïga, n'offrait pas un feedback satisfaisant côté académique.

Conclusions

L'objectif de ce travail était d'écrire une application web permettant de visualiser différents types de fichier tels que des PDF ou des images, directement dans un navigateur web. Pour des questions de performance, une partie du traitement de ces fichiers devaient être fait sur le serveur. De plus, notre application devait respecter le fonctionnement de son prédécesseur en y apportant des améliorations technologiques.

Si côté frontend nous n'avons pas rencontré de problèmes particuliers, côté backend en revanche, nous nous sommes vite rendu compte que sous Python, il n'existait pas de librairie couvrant toutes les fonctionnalités des traitements des fichiers PDF. Aucune de ces librairies ne couvraient tous les besoins de notre application.

Pour pallier à ce problème, nous avons décidé d'utiliser Poppler comme pour la précédente version, mais en remplaçant le wrapping existant utilisant Swig par une nouvelle version en Cython. Cette partie du travail, particulièrement délicate, a demandé un effort conséquent pour résoudre tous les problèmes. Une fois cette solution validée, nous avons pu rattraper le retard accumulé.

A ce jour, nous avons un prototype fonctionnel qui nous permet de visualiser des fichiers d'images et de PDF, de faire des recherches textuelles et permet la navigation via la table des matières et les miniatures.

Un fois le prototype testé, il restera à effectuer d'éventuelles corrections et à étendre l'application à de nouveaux type de documents tels que l'audio et la vidéo. Cette extension devrait être aisée car l'application a été écrite pour être facilement étendue.

Références

Adaptateur (patron de conception). (2018, 4 26). Récupéré sur fr.wikipedia.org: https://fr.wikipedia.org/w/index.php?title=Adaptateur_(patron_de_conception)&oldid=147881836

Advanced Vector Extensions. (2018, 6 28). Retrieved from fr.wikipedia.org: https://fr.wikipedia.org/w/index.php?title=Advanced_Vector_Extensions&oldid=149901943

Agriya. (2017, 5 15). *Pros and Cons of Vue.js framework*. Récupéré sur agriya.com: https://www.agriya.com/blog/pros-and-cons-of-vue-js-framework/

Angular. (2018, 7 28). Retrieved from fr.wikipedia.org: https://fr.wikipedia.org/w/index.php?title=Angular&oldid=150774130

Ant Design of Angular. (s.d.). Récupéré sur ng.ant.design: https://ng.ant.design/docs/introduce/en

Belval. (s.d.). *pdf2image*. Récupéré sur github.com: https://github.com/Belval/pdf2image

Comparison. (s.d.). Récupéré sur fr.vuejs.org: https://fr.vuejs.org/v2/guide/comparison.html

Cython. (2018, 3 22). Retrieved from fr.wikipedia.org: https://fr.wikipedia.org/w/index.php?title=Cython&oldid=146694229

Cython. (s.d.). *Using C++ in Cython — Cython 0.29a0 documentation*. Récupéré sur docs.cython.org: http://docs.cython.org/en/latest/src/userguide/wrapping_CPlusPlus.html#simplified-wrapping-with-default-constructor

Cython: The most widely used Python to C compiler. (2018, 7). Récupéré sur github.com: https://github.com/cython/cython

disjfa. (2017, 1 20). *Using vue.js in existing websites, the easy way*. Récupéré sur medium.com: https://medium.com/@disjfa/using-vue-js-in-existing-websites-the-easy-way-d46cd1f0c945

DMTN-013: Wrapping C++ with Cython. (2017, 08 07). Récupéré sur https://dmtn-013.lsst.io/

Echessa, J. (2017, Avril 20). *Image Processing in Python with Pillow*. Récupéré sur auth0.com: https://auth0.com/blog/image-processing-in-python-with-pillow/

GitBooks. (s.d.). *4 Architecture*. Récupéré sur unbug.gitbooks.io: https://unbug.gitbooks.io/react-native-training/content/4_architecture.html

GNU. (2018, 3 13). *AllCompatibility*. Récupéré sur Gnu.org: https://www.gnu.org/licenses/gpl-faq.html#AllCompatibility

GNU. (2018, 02 11). *GPL Incompatible Licenses*. Récupéré sur Gnu.org: https://www.gnu.org/licenses/license-list.fr.html#GPLIncompatibleLicenses

Google. (s.d.). *Angular - What is Angular?* Récupéré sur angular.io: https://angular.io/docs

IntegratingPythonWithOtherLanguages - Python Wiki. (s.d.). Récupéré sur https://wiki.python.org/moin/IntegratingPythonWithOtherLanguages

Invenio. (s.d.). Récupéré sur en.wikipedia.org: https://en.wikipedia.org/wiki/Invenio

JSX. (s.d.). Récupéré sur jsx.github.io: https://jsx.github.io/

Levia3. (2017, 11 18). *Pdfminer docs*. Récupéré sur media.readthedocs.org: https://media.readthedocs.org/pdf/pdfminer-docs/latest/pdfminer-docs.pdf

Licence Apache. (2018, 3 28). Retrieved from fr.wikipedia.org: https://fr.wikipedia.org/w/index.php?title=Licence_Apache&oldid=146871455

Licence BSD. (2017, 1 27). Retrieved from fr.wikipedia.org: https://fr.wikipedia.org/w/index.php?title=Licence_BSD&oldid=134022823

Licence MIT. (2018, 6 4). Retrieved from fr.wikipedia.org: https://fr.wikipedia.org/w/index.php?title=Licence_MIT&oldid=149216437

Licence publique générale GNU. (2018, 3 2). Retrieved from /fr.wikipedia.org: https://fr.wikipedia.org/w/index.php?title=Licence_publique_g%C3%A9n%C3%A9rale_GNU&oldid=145987251

Lynch, M. (2016, 5). Using Wand to extract images from PDFs in python. Récupéré sur http://mikelynchgames.com/software-development/using-wand-to-extract-pngs-from-pdfs/

Martin. (2018, Février 6). Récupéré sur https://technologiemedia.net: https://technologiemedia.net/2018/02/06/quel-est-le-navigateur-web-le-plus-populaire-pour-janvier-2018/

Mcnamara, T. (s.d.). *Slate*. Récupéré sur github.com: https://github.com/timClicks/slate

Medioni, L. (2014, 03 13). *Ecriture de liaisons C++ pour Python (Logilab.org)*. Récupéré sur logilab.org: https://www.logilab.org/blogentry/228982

Minhee, H. (s.d.). *Wand — Wand 0.4.4*. Récupéré sur docs.wand-py.org: http://docs.wand-py.org/en/0.4.4/

Minhee, H. (s.d.). *Wand Image*. Récupéré sur http://docs.wand-py.org: http://docs.wand-py.org/en/0.4.4/wand/image.html

Nelson, J. K. (2017, 8 17). *How to integrate React into an existing app?* Récupéré sur reactarmory.com: https://reactarmory.com/answers/how-to-integrate-react-into-existing-app

OpenCV. (2018, 6 30). Retrieved from fr.wikipedia.org: https://fr.wikipedia.org/w/index.php?title=OpenCV&oldid=149968323

Parsing PDF for Fun And Profit (indeed in Python) \textbar Ivanovo. (2016, 2 27). Retrieved from http://zderadicka.eu: http://zderadicka.eu/parsing-pdf-for-fun-and-profit-indeed-in-python/

pdf2image 0.1.0. (s.d.). Récupéré sur pypi.python.org: https://pypi.python.org/pypi/pdf2image/0.1.0

pdfminer3k 1.3.1. (s.d.). Récupéré sur pypi.python.org: https://pypi.python.org/pypi/pdfminer3k/

Pillow. (s.d.). *Pillow*. Récupéré sur pillow.readthedocs.io: https://pillow.readthedocs.io/en/latest/

Point par pouce. (2018, 5 25). Récupéré sur fr.wikipedia.org: https://fr.wikipedia.org/wiki/Point_par_pouce

Poppler 0.18 (0.63.0) - Poppler 0.18. (s.d.). Récupéré sur lazka.github.io: https://lazka.github.io/pgi-docs/#Poppler-0.18

Poppler. (s.d.). *Poppler*. Récupéré sur https://poppler.freedesktop.org/

PyBindGen Tutorial — PyBindGen 0.17.0 documentation. (s.d.). Récupéré sur http://pybindgen.readthedocs.io: http://pybindgen.readthedocs.io/en/latest/tutorial/

Python converting PDF to Image. (2013, Juillet 13). Récupéré sur garmoncheg.blogspot.ch: http://garmoncheg.blogspot.ch/2013/07/python-converting-pdf-to-image.html

React JS. (s.d.). Récupéré sur en.wikipedia.org: https://en.wikipedia.org/wiki/React_(JavaScript_library)

Reitz, K. (s.d.). *Image manipulation*. Récupéré sur docs.python-guide.org: http://docs.python-guide.org/en/latest/scenarios/imaging/

RERO. (2005, 11 16). *RERO DOC*. Récupéré sur www.rero.ch: https://www.rero.ch/page.php?section=infos&pageid=rerodoc

RERO. (2017, 8 16). *RERO: Informations générales*. Récupéré sur rero.ch: https://www.rero.ch/page.php?section=infos&pageid=rero_info

Single instruction multiple data. (2018, 1 23). Retrieved from fr.wikipedia.org: https://fr.wikipedia.org/w/index.php?title=Single_instruction_multiple_data&oldid=144758566

SIP (software). (2018, 1 7). Retrieved from en.wikipedia.org: https://en.wikipedia.org/w/index.php?title=SIP_(software)&oldid=819136908

Slate 0.5.2. (s.d.). Récupéré sur pypi.python.org: https://pypi.python.org/pypi/slate

Smith, B. (2016, 9 14). *Guide rapide de la GPLv3*. Retrieved from gnu.org: https://www.gnu.org/licenses/quick-guide-gplv3.fr.html

SWIG. (2017, 7 29). Retrieved from fr.wikipedia.org: https://fr.wikipedia.org/w/index.php?title=SWIG&oldid=139326117

TechMagic. (2018, 3 16). *ReactJS vs Angular5 vs Vue.js—What to choose in 2018?* Récupéré sur medium.com: https://medium.com/@TechMagic/reactjs-vs-angular5-vs-vue-js-what-to-choose-in-2018-b91e028fa91d

Thomas, J. (s.d.). *Alternative to Bootstrap*. Récupéré sur bulma.io: https://bulma.io/alternative-to-bootstrap/

TypeScript — Wikipédia. (2018, 7 23). Récupéré sur fr.wikipedia.org: https://fr.wikipedia.org/wiki/TypeScript

Uploadcare. (2017, Juin 20). *The fastest production-ready image resize out there*. Récupéré sur blog.uploadcare.com: https://blog.uploadcare.com/the-fastest-production-ready-image-resize-out-there-part-0-7c974d520ad9

Uploadcare. (s.d.). *Pillow perf*. Récupéré sur github.com: https://github.com/python-pillow/pillow-perf

Uploadcare. (s.d.). *Pillow SIMD*. Récupéré sur github.com: https://github.com/uploadcare/pillow-simd

vfdef. (2017, Octobre 13). *PIL vs Opencv | Kaggle*. Récupéré sur Kaggle.com: https://www.kaggle.com/vfdev5/pil-vs-opencv

Vorbach, P. (s.d.). *npm-stat: download statistics for NPM packages*. Récupéré sur https://npm-stat.com/

VueJS. (s.d.). *Building Large-Scale Apps - vue.js*. Récupéré sur v1.vuejs.org: https://v1.vuejs.org/guide/application.html

Working with PDF and Word Documents. (s.d.). Récupéré sur automatetheboringstuff.com: https://automatetheboringstuff.com/chapter13/

Zderadicka, I. (2018, 7). *pdfparser: Python binding to libpoppler with focus on text extraction*. Récupéré sur https://github.com/izderadicka/pdfparser

Annexes

Annexe I : Ficher XML généré lors de la déposition d'un nouveau document

```
<?xml version="1.0" encoding="UTF-8"?>
<collection xmlns="http://www.loc.gov/MARC21/slim">
<record>
  <controlfield tag="001">259080</controlfield>
  <controlfield tag="005">20160402122825.0</controlfield>
  <datafield tag="024" ind1="8" ind2=" ">
    <subfield code="a">oai:doc.rero.ch:20160401111647-AI</subfield>
    <subfield code="p">tdhes</subfield>
    <subfield code="p">hesvd</subfield>
    <subfield code="p">dissertation</subfield>
    <subfield code="z">cdu615.8</subfield>
    <subfield code="z">hesav</subfield>
  </datafield>
  <datafield tag="035" ind1=" " ind2=" ">
    <subfield code="a">R008394632</subfield>
  </datafield>
  <datafield tag="041" ind1=" " ind2=" ">
    <subfield code="a">fre</subfield>
  </datafield>
  <datafield tag="080" ind1=" " ind2=" ">
    <subfield code="a">615.8</subfield>
  </datafield>
  <datafield tag="100" ind1=" " ind2=" ">
    <subfield code="a">Lacheteau, Grégoire</subfield>
  </datafield>
  <datafield tag="245" ind1=" " ind2=" ">
    <subfield code="a">Les "Transversus Abdominis" et la contraction
asymétrique</subfield>
    <subfield code="9">fre</subfield>
    <subfield code="b">quels patterns d'activation lors de déstabilisation spinale
induite par les membres supérieurs?</subfield>
  </datafield>
  <datafield tag="300" ind1=" " ind2=" ">
    <subfield code="a">34 p.</subfield>
  </datafield>
 <datafield tag="502" ind1=" " ind2=" ">
    <subfield code="a">Mémoire de bachelor : Haute Ecole de Santé Vaud,
2015</subfield>
    <subfield code="9">2015</subfield>
  </datafield>
```

Figure 43: Exemple fichier XML, informations du document partie 1

```
<datafield tag="502" ind1=" " ind2=" ">
    <subfield code="a">Mémoire de bachelor : Haute Ecole de Santé Vaud,
2015</subfield>
    <subfield code="9">2015</subfield>
  </datafield>
  <datafield tag="520" ind1=" " ind2=" ">
    <subfield code="a">Introduction : Actuellement le traitement …subfield>
    <subfield code="9">fre</subfield>
  </datafield>
  <datafield tag="520" ind1=" " ind2=" ">
    <subfield code="a">Introduction: Currently the treatment …</subfield>
    <subfield code="9">eng</subfield>
  </datafield>
  <datafield tag="695" ind1=" " ind2=" ">
    <subfield code="a">EMG ; contrôle moteur ; contraction ; activation</subfield>
    <subfield code="9">fre</subfield>
  </datafield>
  <datafield tag="695" ind1=" " ind2=" ">
    <subfield code="a">motor control ; contraction ; activation ;
activity</subfield>
    <subfield code="9">eng</subfield>
  </datafield>
  <datafield tag="695" ind1=" " ind2=" ">
    <subfield code="a">transversus abdominis</subfield>
    <subfield code="9">lat</subfield>
  </datafield>
  <datafield tag="700" ind1=" " ind2=" ">
    <subfield code="a">Lerch, Maude</subfield>
  </datafield>
  <datafield tag="700" ind1=" " ind2=" ">
    <subfield code="a">Schnyder, David</subfield>
  </datafield>
  <datafield tag="700" ind1=" " ind2=" ">
    <subfield code="a">Degache, Francis</subfield>
    <subfield code="e">Dir.</subfield>
  </datafield>
  <datafield tag="856" ind1="4" ind2=" ">
    <subfield code="f">HESAV_TB_Lacheteau_2015.pdf</subfield>
    <subfield code="q">application/pdf</subfield>
    <subfield code="s">1349513</subfield>
    <subfield
code="u">http://doc.rero.ch/record/259080/files/HESAV_TB_Lacheteau_2015.pdf</subfie
ld>
    <subfield code="y">order:1</subfield>
    <subfield code="z">Texte intégral</subfield>
  </datafield>
  <datafield tag="919" ind1=" " ind2=" ">
    <subfield code="a">Haute Ecole de Santé Vaud</subfield>
    <subfield code="b">Lausanne</subfield>
    <subfield code="d">doc.support@rero.ch</subfield>
  </datafield>
  <datafield tag="980" ind1=" " ind2=" ">
    <subfield code="a">DISSERTATION</subfield>
    <subfield code="b">HESAV</subfield>
    <subfield code="f">DISS_BACHELOR</subfield>
  </datafield>
```

Figure 44: Exemple fichier XML, informations du document partie 2

Annexe II : Informations du système

Les différents tests ont été effectués sur une machine Ubuntu 16.04 LTS dont le disque dur est non SDD.

En image, les informations concernant le système.

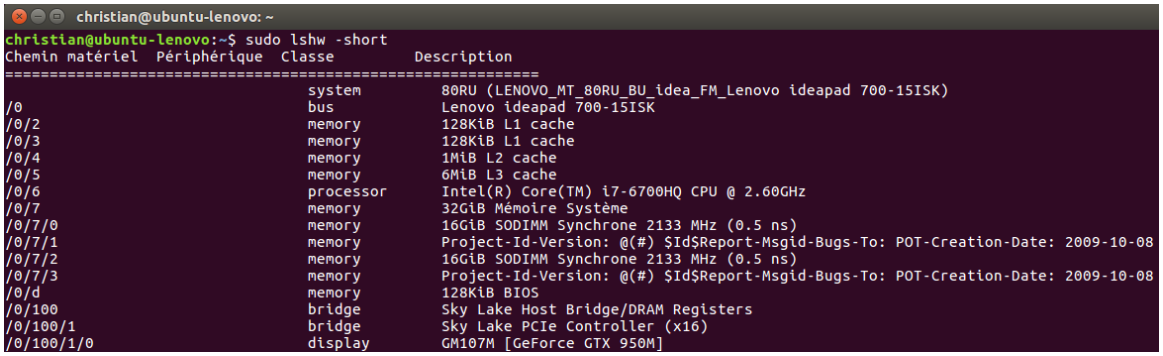

Figure 45: Informations du système

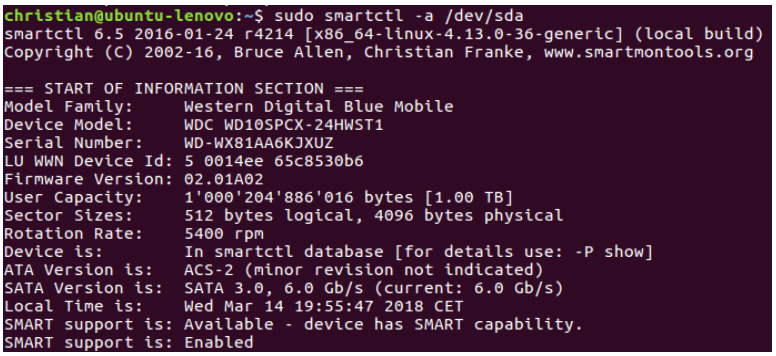

Figure 46: Informations du disque de stockage

```
christian@ubuntu-lenovo:~$ lscpu
Architecture:            x86_64
Mode(s) opératoire(s) des processeurs :32-bit, 64-bit
Byte Order:              Little Endian
CPU(s):                  8
On-line CPU(s) list:     0-7
Thread(s) par cœur : 2
Cœur(s) par socket : 4
Socket(s):               1
Nœud(s) NUMA :       1
Identifiant constructeur :GenuineIntel
Famille de processeur :6
Modèle :             94
Model name:              Intel(R) Core(TM) i7-6700HQ CPU @ 2.60GHz
Révision :           3
Vitesse du processeur en MHz :2600.000
CPU max MHz:             3500.0000
CPU min MHz:             800.0000
BogoMIPS:                5184.00
Virtualisation :      VT-x
Cache L1d :              32K
Cache L1i :              32K
Cache L2 :               256K
Cache L3 :               6144K
NUMA node0 CPU(s):       0-7
```

Figure 47: Informations CPU

Annexe III : Manipulations d'images, script de test 1

```
from PIL import Image
from wand.image import Image as wnd
import time
import cv2 as cv
import numpy as np
import timeit
import click

@click.command()
@click.option('--file', default='4252x2835.jpg')
@click.option('--loops', default=1)
def StartTest(file, loops):
    global inputFile
    inputFile = file
    print("\n------------ Image Manipulation tests ----------\n")
    print("Test are done with image size:", file, "with", loops, "loops")
    print("Pillow: ")
    print("Load :", timeit.timeit(pillowLoad, number = loops))
    print("Rotation:", timeit.timeit(pillowTestRotation,number = loops))
    print("Transpose:", timeit.timeit(pillowTestTranspose,number = loops))
    print("Resize:", timeit.timeit(pillowTestResize,number = loops))

    print("\nOpenCV: ")
    print("Load :" , timeit.timeit(opencvLoad ,number = loops))
    print("Rotation :", timeit.timeit(opencvTestRotation,number = loops))
    print("Transpose :", timeit.timeit(opencvTestTranspose,number = loops))
    print("Resize :", timeit.timeit(opencvTestResize,number = loops))


    print("\nWand: ")
    print("Load :", timeit.timeit(wandLoad, number = loops))
    print("Rotation:" , timeit.timeit(wandTestRotation,number = loops))
    print("Transpose:" , timeit.timeit(wandTestTranspose,number = loops))
    print("Resize:" , timeit.timeit(wandTestRotation,number = loops))
```

Figure 48: Script comparaison Pillow, OpenCV, Wand. Partie 1

```
#---------- Load image --------------
def pillowLoad():
    global im
    im = Image.open( inputFile )

def opencvLoad():
    global img
    img = cv.imread( inputFile )

def wandLoad():
    with wnd (filename = inputFile) as image:
        pass
#---------- Rotation --------------
def pillowTestRotation():
    im.rotate(45)

def opencvTestRotation():
    image_center = tuple(np.array(img.shape[1::-1]) / 2)
    cv.getRotationMatrix2D(image_center, 45 , 1.0)

def wandTestRotation():
    with wnd (filename = inputFile) as image:
        image.rotate(45)

#---------- Transpose --------------
def pillowTestTranspose():
    im.transpose(Image.FLIP_LEFT_RIGHT)

def opencvTestTranspose():
    cv.flip(img, flipCode=1)

def wandTestTranspose():
    with wnd (filename = inputFile) as image:
        image.flop()

#---------- Resize --------------
def pillowTestResize():
    im.resize((512, 512), Image.CUBIC)

def opencvTestResize():
    cv.resize(img, dsize=(512, 512), interpolation=cv.INTER_CUBIC)

def wandTestResize():
    with wnd (filename = inputFile) as image:
        image.resize(512,512)

#---------- MAIN --------------
if __name__ == '__main__':
    StartTest()
```

Figure 49: Script comparaison Pillow, OpenCV, Wand. Partie 2

Voici les tableaux récapitulatifs de mesures, en vert les meilleurs temps :

1 itération		24x24	320x200	2048x1280	4252x2835
Pillow	**Load**	0.009550497001	0.0095665049993	**0.0095049169976**	**0.009544462001**
	Rotation	0.0002612200005	0.0024399469984	0.0926235570004	0.2943923639977
	Transpose	2.449500243 e-05	**0.000124017999**	**0.0059743029996**	**0.0304179099985**
	Resize	0.0050385719987	0.0051905859982	0.0272875580012	0.1063399090016
OpenCV	**Load**	**0.000228867997**	**0.0012197069991**	0.064792687997	0.152128900001
	Rotation	**5.391000013 e-05**	**4.982199970 e-05**	**5.468000017 e-05**	**5.675500142 e-05**
	Transpose	**2.327800029 e-05**	0.0001274179994	0.005959328002	0.0361070510007
	Resize	0.004317543996	**0.0008886729992**	**0.0036292289987**	**0.0053225820010**
Wand	**Load**	0.002888156999	0.0031753029979	0.063941663	0.1511682239979
	Rotation	0.07964403	0.0948829559965	0.574372605999	2.2538460299983
	Transpose	0.0006719820012	0.001856625	0.0703646790025	0.178612718002
	Resize	**0.002545457999**	0.01915818199	0.528262130999	2.273007554998

Tableau 11: Résultats du temps d'exécution des manipulations images, test1 (1 itération)

Résultats des mesures avec dix itérations :

10 itérations		24x24	320x200	2048x1280	4252x2835
Pillow	Load	0.0149410670019	0.0150197830007	**0.0119079030009**	**0.0120229860003**
	Rotation	0.0006193810004	0.0050046050018	0.1948593910019	1.1275190400010
	Transpose	0.0001156430007	**0.0008425250016**	**0.0256922189983**	**0.3036562779998**
	Resize	0.0298977029997	0.0404929639989	0.2768460020015	1.0409730489991
OpenCV	Load	**0.0008145739993**	**0.0105145049965**	0.6254963999999	1.5325653750005
	Rotation	**0.0001183690001**	**0.0001350950005**	**0.0001226099993**	**0.0001321189993**
	Transpose	**4.477999755 e-05**	0.0011059559983	0.0637019169989	0.3550642070003
	Resize	**0.0049045019986**	**0.0087662499972**	**0.0325031659995**	**0.0491756150004**
Wand	Load	0.0076179489988	0.0169390549999	0.5475980549999	1.4940861110007
	Rotation	0.1024176000028	0.2427588929967	5.1546000829985	26.354042486000
	Transpose	0.0058516169992	0.015958928001	0.6882253740004	1.9049501680019
	Resize	0.0263484979986	0.1875683020007	5.0719772029988	36.776405835000

Tableau 12: Résultats du temps d'exécution des manipulations images, test1 (10 itérations)

Annexe IV : Manipulations d'images, script de test 2

```python
from PIL import Image
import time
import cv2 as cv
import numpy as np
import timeit
import click

@click.command()
@click.option('--file', default='4252x2835.jpg')
@click.option('--loops', default=1)
def StartTest(file, loops):
    global inputFile
    inputFile = file
    print("\n------------ Image Manipulation tests ----------\n")
    print("Test are done with image size:", inputFile, "with", loops, "loops")
    print("Pillow-SIMD: ")
    print("Load :", timeit.timeit(pillowLoad, number = loops))
    print("Rotation:", timeit.timeit(pillowTestRotation,number = loops))
    print("Transpose:", timeit.timeit(pillowTestTranspose,number = loops))
    print("Resize:", timeit.timeit(pillowTestResize,number = loops))

    print("\nOpenCV: ")
    print("Load :" , timeit.timeit(opencvLoad ,number = loops))
    print("Rotation :", timeit.timeit(opencvTestRotation,number = loops))
    print("Transpose :", timeit.timeit(opencvTestTranspose,number = loops))
    print("Resize :", timeit.timeit(opencvTestResize,number = loops))


#---------- Rotation --------------
def pillowTestRotation():
    im.rotate(45)

def opencvTestRotation():
    image_center = tuple(np.array(img.shape[1::-1]) / 2)
    cv.getRotationMatrix2D(image_center, 45 , 1.0)

#---------- Transpose --------------
def pillowTestTranspose():
    im.transpose(Image.FLIP_LEFT_RIGHT)

def opencvTestTranspose():
    cv.flip(img, flipCode=1)

#---------- Resize --------------
def pillowTestResize():
    im.resize((512, 512), Image.CUBIC)

def opencvTestResize():
    cv.resize(img, dsize=(512, 512), interpolation=cv.INTER_CUBIC)


#---------- MAIN --------------
if __name__ == '__main__':
    StartTest()
```

Figure 50: Script comparaison Pillow-SIMD et OpenCV

```
(ImageSIMD2) christian@ubuntu-lenovo:~/virtenvs/ImageSIMD2$ python myscript.py -
-file=4252x2835.jpg --loops=10

------------ Image Manipulation tests ----------

Test are done with image size: 4252x2835.jpg with 10 loops
Pillow-SIMD:
Load : 0.00808052800130099
Rotation: 1.0103967110007943
Transpose: 0.2711609599973599
Resize: 0.20892334400195978

OpenCV:
Load : 1.5195958990007057
Rotation : 0.00015472499944735318
Transpose : 0.351318633001938
Resize : 0.049165017997438554
```

Figure 51: Exemple de résultat pour la manipulation d'images, test 2

Voici les tableaux récapitulatifs de mesures, en vert les meilleurs temps :

1 itération		24x24	320x200	2048x1280	4252x2835
Pillow-SIMD	**Load**	0.0090839839976	0.0073427510033	0.0069433040007	0.0070172370033
	Rotation	0.0001884060002	0.001216789001	0.0648676420023	0.1925693369994
	Transpose	1.807299809 e-05	6.858699998 e-05	0.0053394729984	0.0265680390002
	Resize	0.0011694140011	0.0015068629982	0.0056753379976	0.0231710599982
	Total	**0.0104598769969**	**0.0101349900024**	**0.082825756999**	**0.2493256730011**
OpenCV	**Load**	0.0001928450001	0.0012984360009	0.0648303960006	0.149773114997
	Rotation	4.851300036 e-05	4.91249993 e-05	7.405499854 e-05	5.717900057 e-05
	Transpose	2.080699778 e-05	0.0001515149997	0.0055723239966	0.0350508310002
	Resize	0.0012252470005	0.0012322980001	0.0038282579989	0.0052217349984
	Total	**0.0014874119987**	**0.002731374**	**0.0743050329946**	**0.1901028599961**

Tableau 13: Résultats manipulations images test2 (1 itération)

10 itérations		24x24	320x200	2048x1280	4252x2835
Pillow-SIMD	**Load**	0.0100406129968	0.0082306270014	0.0076092820017	0.0079090530016
	Rotation	0.0004518809982	0.0027846479970	0.1667294129983	0.9975928349995
	Transpose	7.031200220 e-05	0.0005661610011	0.0202805269982	0.2666300960008
	Resize	0.0082233339999	0.0108324979992	0.056536969001	0.2086232600013
	Total	**0.0187861399971**	**0.0224139339987**	**0.2511561909992**	**1.4807552440032**
OpenCV	**Load**	0.0008074019970	0.010573903997	0.6229886580003	1.5195488860008
	Rotation	0.0001162909975	0.0001218040015	0.0001238469994	0.0001275699978
	Transpose	4.431699926 e-05	0.0011381290023	0.0561764359990	0.3473758559994
	Resize	0.0047619830002	0.0094070289997	0.0326111260001	0.0490693499996
	Total	**0.0057299929939**	**0.0212408660005**	**0.7119000669988**	**1.9161216619976**

Tableau 14: Résultats manipulations images test2 (10 itérations)

Annexe V : PDF vers image, script de test

```
from pdf2image import convert_from_path
from PIL import Image
from wand.image import Image as wnd
import time
import subprocess

#---------- Timer --------------
def time_usage(func):
    def wrapper(*args, **kwargs):
        beg_ts = time.time()
        retval = func(*args, **kwargs)
        end_ts = time.time()
        print("Elapsed time: %f\n" % (end_ts - beg_ts))
        return retval
    return wrapper

#---------- Fonctions --------------
@time_usage
def pdf2image():
    image = convert_from_path('test.pdf', dpi=300, fmt='jpg')

@time_usage
def wand():
    img= wnd(filename="test.pdf[0]", resolution=300)
    #img.save(filename="results/wand.jpg")

@time_usage
def pdftoppm():
    test = subprocess.Popen(["pdftoppm","-jpeg","-f","1", "-singlefile", "-r", "300",
"test.pdf", "result_pdftoppm"],stdout=subprocess.PIPE)
    output = test.communicate()[0]
```

Figure 52: Script du rendu PDF vers une image

Annexe VI : Extraction du texte depuis un PDF selon une page définie

```
from PIL import Image
import time
from slate import PDF
import PyPDF2
from pdfminer.pdfparser import PDFParser, PDFDocument
from pdfminer.pdfinterp import PDFResourceManager, PDFPageInterpreter
from pdfminer.converter import PDFPageAggregator
from pdfminer.layout import LAParams, LTTextBox, LTTextLine


#Variables
path = 'test.pdf'

#---------- Timer --------------
def time_usage(func):
    def wrapper(*args, **kwargs):
        beg_ts = time.time()
        retval = func(*args, **kwargs)
        end_ts = time.time()
        print("Elapsed time: %f\n" % (end_ts - beg_ts))
        return retval
    return wrapper

@time_usage
def slate():
    with open(path, 'rb') as f:
        doc = PDF(f)
        #print (doc[8])


@time_usage
def pypdf2():
    pdfFileObj = open(path, 'rb')
    pdfFile = PyPDF2.PdfFileReader(pdfFileObj)
    if pdfFile.isEncrypted:
        try:
            pdfFile.decrypt('')
            print('File Decrypted (PyPDF2)')
        except:
            command = ("cp "+ filename +
                " temp.pdf; qpdf --password='' --decrypt temp.pdf " + filename
                + "; rm temp.pdf")
            os.system(command)
            print('File Decrypted (qpdf)')
            fp = open(filename)
            pdfFile = PdfFileReader(fp)
    else:
        print('File Not Encrypted')

    pageObj = pdfFile.getPage(8)
    text_extracted = pageObj.extractText()
    #print(text_extracted)
```

Figure 53: Script d'extraction du texte depuis un PDF, partie 1

```
@time_usage
def pdfminer():
    fp = open(path, 'rb')
    parser = PDFParser(fp)
    doc = PDFDocument()
    parser.set_document(doc)
    doc.set_parser(parser)
    doc.initialize('')
    rsrcmgr = PDFResourceManager()
    laparams = LAParams()
    device = PDFPageAggregator(rsrcmgr, laparams=laparams)
    interpreter = PDFPageInterpreter(rsrcmgr, device)
    # Process each page contained in the document.
    for pageNumber, page in enumerate(doc.get_pages()):
        if pageNumber == 8:
            interpreter.process_page(page)
            layout = device.get result()
            #for lt_obj in layout:
                #if isinstance(lt_obj, LTTextBox) or isinstance(lt_obj, LTTextLine):
                    #print(lt_obj.get_text())
```

Figure 54: Script d'extraction du texte depuis un PDF, partie 2

Annexe VII : Extraction de texte avec localisation

```
from PIL import Image
import time
from pdfminer.pdfparser import PDFParser, PDFDocument
from pdfminer.pdfinterp import PDFResourceManager, PDFPageInterpreter
from pdfminer.converter import PDFPageAggregator
from pdfminer.layout import LAParams, LTTextBox, LTTextLine, LTText, LTFigure, LTAnon, LTChar
import click

#Variables
bboxList = []
tmpListe = []
matches = [[],[]]

#---------- Timer --------------
def time_usage(func):
    def wrapper(*args, **kwargs):
        beg_ts = time.time()
        retval = func(*args, **kwargs)
        end_ts = time.time()
        print("Elapsed time: %f\n" % (end_ts - beg_ts))
        return retval
    return wrapper

#---------- pdfminer --------------
@click.command()
@click.option('--path', default="doc.pdf")
@click.option('--string', default="photosynthesis")
@click.option('--p', default=-1)
@click.option('--mode', default="search")
@time_usage
```

Figure 55: Script localisation du texte dans un PDF, partie 1

```
def pdfminer(path,string,p,mode):
    global stringToFind
    stringToFind = " "+string.lower()+" "
    stringToFind = stringToFind.replace("  "," ")
    fp = open(path, 'rb')
    parser = PDFParser(fp)
    doc = PDFDocument()
    parser.set_document(doc)
    doc.set_parser(parser)
    doc.initialize('')
    rsrcmgr = PDFResourceManager()
    laparams = LAParams()
    device = PDFPageAggregator(rsrcmgr, laparams=laparams)
    interpreter = PDFPageInterpreter(rsrcmgr, device)
    if mode == 'toc':
        outlines = doc.get_outlines()
        for (level,title,dest,a,se) in outlines:
            print (level, title)
    elif mode == 'search':
        for pageNumber, page in enumerate(doc.get_pages()):
            if p == -1:
                interpreter.process_page(page)
                layout = device.get_result()
                parse_layout(layout,pageNumber)
            elif pageNumber == p:
                interpreter.process_page(page)
                layout = device.get_result()
                parse_layout(layout,p)
        print(matches)

def parse_layout(layout,pageNumber):
    for lt_obj in layout:
        if isinstance(lt_obj, LTTextBox):
            parse_layout(lt_obj,pageNumber)  # Recursive
        elif isinstance(lt_obj, LTTextLine) or isinstance(lt_obj, LTText):
            if stringToFind in lt_obj.get_text().lower():
                temp = ""
                bboxTemp = []
                tmpListe = []
                bboxList = []
                isFirsCaracter = True
                for char in lt_obj:
                    if char.get_text() in " .,:!?;‟(){[]\n":
                        if temp  != "":
                            tmpListe.append(temp)
                            bboxList.append(bboxTemp)
                        temp = ""
                        bboxTemp = []
                        isFirsCaracter = True
                    else:
                        temp+=char.get_text().lower()
                        if isFirsCaracter is True:
                            bboxTemp = list(char.bbox)
                            isFirsCaracter = False
                        else:
                            bboxTemp[2] = char.bbox[2]
                index = tmpListe.index(stringToFind.replace(" ",""))
                matches[0].append(bboxList[index])
                matches[1].append(pageNumber)
        elif isinstance(lt_obj, LTFigure):
            parse_layout(lt_obj,pageNumber)  # Recursive

#---------- MAIN --------------
if __name__ == '__main__':
    print("---- pdfminer  ---- ")
    pdfminer()
```

Figure 56: Script localisation du texte dans un PDF, partie 2

Annexe VIII : Fonction extraction table des matières (wrapping de Poppler)

```
def getToc(self):
      cdef:
        Object* outline
        Object node
        Dict* dict

      childs = PyList_New(0)
      outline = self.doc.getCatalog().getOutline()
      if (outline != NULL and outline.isDict()):
          self.newOutlineLevel(<Object* > outline ,<Catalog*> self.doc.getCatalog(), childs, 1)
      return childs

  cdef void newOutlineLevel(self, Object* node, Catalog* catalog, list myList, int level):
    cdef:
        Object next
        Object curr
        GBool atLeastOne = False
        int page_number = -1
        Unicode *title = NULL
        int titleLen = 0
        GooString* s = NULL
        int i
        GooString *linkName = NULL
        Object dest
        Object obj_title
        Object obj_curr
        Ref pageref
        LinkDest *linkdest
        LinkGoTo *action
        LinkGoTo *link

    curr = node.dictLookup("First", 0)
    if(curr.isDict()):
      while True:
        page_number = -1
        obj_title =  curr.dictLookup("Title", 0)
        if (obj_title.isString() and obj_title.isNull() == False):
          s = <GooString *>obj_title.getString()
          if ((s.getChar(0) & 0xff) == 0xfe and (s.getChar(1) & 0xff) == 0xff):
            titleLen = (s.getLength() - 2) / 2
            title = <Unicode *>PyObject_Malloc(titleLen * sizeof(Unicode))
            for i in range(0, titleLen):
              title[i] = ((s.getChar(2 + 2*i) & 0xff) << 8) or (s.getChar(3 + 2*i) & 0xff)
          else:
            titleLen = s.getLength()
            title = <Unicode *>PyObject_Malloc(titleLen * sizeof(Unicode))
            for i in range(0, titleLen):
              title[i] = pdfDocEncoding[s.getChar(i) & 0xff]
        else:
          titleLen = 0
          #PyMem_Free(&obj_title)
          break
        #PyMem_Free(&obj_title)
        if (titleLen == 0):
          titleLen = 1
          title = <Unicode *>PyObject_Malloc(titleLen * sizeof(Unicode))
          title[0] = '\0'
```

Figure 57: Fonction pour récupération table des matières, partie 1

```
#get corresponding link
dest = curr.dictLookup("A", 0)
if (dest.isNull() == False):
  action = <LinkGoTo *>LinkAction.parseAction(&dest, NULL)
  if(action and action.isOk()):
    linkdest = NULL
    if (action.getDest()!=NULL):
      linkdest=action.getDest().copy()
    elif (action.getNamedDest()!=NULL):
      linkdest=catalog.findDest(action.getNamedDest())
    #PyMem_Free(&action)
    if (linkdest):
      if (linkdest.isPageRef()):
        pageref = linkdest.getPageRef()
        page_number = catalog.findPage(pageref.num, pageref.gen)
      else:
        page_number = linkdest.getPageNum()
      #PyMem_Free(&linkdest)
  #PyMem_Free(&dest)
dest = curr.dictLookup("Dest", 0)
if (dest.isNull() == False):
  link = new LinkGoTo(&dest)
  linkdest=NULL
  if (link.getDest()!=NULL):
    linkdest=link.getDest().copy()
  elif (link.getNamedDest()!=NULL):
    linkdest=catalog.findDest(link.getNamedDest())
  #PyMem_Free(&link)
  if (linkdest):
    if (linkdest.isPageRef()):
      pageref=linkdest.getPageRef()
      page_number=catalog.findPage(pageref.num,pageref.gen)
    else:
      page_number=linkdest.getPageNum()
    #PyMem_Free(&linkdest)
  #PyMem_Free(&dest)

local_dic =  PyDict_New()
childs = PyList_New(0)

self.newOutlineLevel(&curr, <Catalog*>catalog, childs, level+1)
PyDict_SetItemString(local_dic, "label", (PyUnicode_FromUnicode(<Py_UNICODE *>title, titleLen)))
if (page_number < 1 or page_number > catalog.getNumPages()):
  PyDict_SetItemString(local_dic, "page_number", None)
else:
  PyDict_SetItemString(local_dic, "page_number", PyInt_FromLong(page_number))
if (PyList_Size(childs) > 0):
  PyDict_SetItemString(local_dic, "childs", childs)
PyList_Append(myList, local_dic)
curr = curr.dictLookup("Next",0)
if curr.isDict() == False:
  break
```

Figure 58: Fonction pour récupération table des matières, partie 2

Annexe IX : Fonction récupération métadonnées du document (wrapping de Poppler)

```
def getInfo(self):
  cdef:
    Object info
    Object obj
    Unicode *titleInfo = NULL
    int titleLenght = 0
    GooString* s = NULL
    char *key

  dic =  PyDict_New()
  info = self.doc.getDocInfo()
  if not info.isDict():
    return dic

  info_dict = info.getDict()
  for i in range(0, info_dict.getLength()):
      key = info_dict.getKey(i)
      obj = info_dict.lookup(key, 0)
      if (obj.isString()):
        s = <GooString *>obj.getString()
        if ((s.getChar(0) & 0xff) == 0xfe and (s.getChar(1) & 0xff) == 0xff):
          titleLenght = (s.getLength() - 2) / 2
          titleInfo = <Unicode *>PyObject_Malloc(titleLenght * sizeof(Unicode))
          for i in range(0, titleLenght):
            titleInfo[i] = ((s.getChar(2 + 2*i) & 0xff) << 8) or (s.getChar(3 + 2*i) & 0xff)
        else:
          titleLenght = s.getLength()
          titleInfo = <Unicode *>PyObject_Malloc(titleLenght * sizeof(Unicode))
          for j in range(0, titleLenght):
            titleInfo[j] = pdfDocEncoding[s.getChar(j) & 0xff]
        if (titleLenght == 0):
          titleLenght = 1
          titleInfo = <Unicode *>PyObject_Malloc(titleLenght * sizeof(Unicode))
          titleInfo[0] = '\0'

        if(titleLenght > 0):
          PyDict_SetItemString(dic, key, PyUnicode_FromUnicode(<Py_UNICODE *>titleInfo, titleLenght))
      #gfree(titleInfo)
    #obj.free();
  return dic
```

Figure 59: Fonction pour récupération métadonnées du PDF

Annexe X : Rendu PDF vers image (wrapping de Poppler)

```
cdef class Image:
  cdef:
      SplashOutputDev *splash
      SplashBitmap *bitmap
      double scale
      Document pdf
      bytes data

  def __cinit__(self, int page, Document pdf, int max_width, int max_height):
      self.pdf = pdf
      self.splash = new SplashOutputDev(splashModeRGB8, 3, False, [255,255,255], True,
splashThinLineDefault, False)
      self.splash.setFontAntialias(True)
      self.splash.setVectorAntialias(True)
      self.splash.startDoc(self.pdf.doc)
      self.scale = self.getOptimalScale(max_width, max_height, page)
      self.pdf.doc.displayPage(<OutputDev*>self.splash, page, 72*self.scale, 72*self.scale, 0, True, True,
False)
      self.bitmap = self.splash.getBitmap()

  def getScale(self):
      return self.scale

  def getBitmap(self):
      data =  <bytes> (<char *>self.bitmap.getDataPtr())[:3*self.getWidth()*self.getHeight()]
      return data

  def getHeight(self):
      return self.bitmap.getHeight()

  def getWidth(self):
      return self.bitmap.getWidth()

  def getOptimalScale(self, max_width, max_height, page_nr):
        """Compute the optimal scale factor."""
        if max_width is None and max_height is None:
            return 1.0
        page_width = self.pdf.doc.getPageMediaWidth(page_nr)
        page_height = self.pdf.doc.getPageMediaHeight(page_nr)
        if(int(self.pdf.doc.getPageRotate(page_nr)) % 180 == 90):
            page_width, page_height = page_height, page_width

        page_ratio = page_height/float(page_width)
        if max_width is None:
            max_width = max_height/page_ratio
        if max_height is None:
            max_height = max_width*page_ratio
        scale = max_width/page_width
        if max_height < (page_height*scale):
            scale = max_height/page_height
        return scale
```

Figure 60: Classe pour rendu PDF vers image

Déclaration de l'auteur

Je déclare, par ce document, que j'ai effectué le travail de Bachelor ci-annexé seul, sans autre aide que celles dûment signalées dans les références, et que je n'ai utilisé que les sources expressément mentionnées. Je ne donnerai aucune copie de ce rapport à un tiers sans l'autorisation conjointe du RF et du professeur chargé du suivi du travail de Bachelor, à l'exception des personnes qui m'ont fourni les principales informations nécessaires à la rédaction de ce travail et que je cite ci-après :

Dominique Genoud

Pereira Christian

Miguel Moreira

Johnny Mariéthoz