



Ecole Polytechnique de l'Université de Tours  
Département Informatique  
64 avenue Jean Portalis  
37200 Tours, France  
Tél. +33 (0)2 47 36 14 14  
[polytech.univ-tours.fr](http://polytech.univ-tours.fr)

## Projet Recherche & Développement 2020-2021

# CESR-DI : outil en ligne pour le balisage stand-off d'un appareil critique



**Entreprise**  
Centre d'Études Supérieures de la Renaissance



**Tuteur entreprise**  
Elena PIERAZZO

**Étudiant**  
Dylan MOTARD (DI5)

**Tuteurs académiques**  
Mohamed SLIMANE  
Jean-Yves RAMEL

# Liste des intervenants

## Entreprise

Centre d'Études Supérieures de la Renaissance  
59, rue Néricault-Destouches - BP 12050  
37020 Tours Cedex 1, France  
[cesr.univ-tours.fr](http://cesr.univ-tours.fr)



Nom	Email	Qualité
Dylan MOTARD	<a href="mailto:dylan.motard@etu.univ-tours.fr">dylan.motard@etu.univ-tours.fr</a>	Étudiant DI5
Mohamed SLIMANE	<a href="mailto:mohamed.slimane@univ-tours.fr">mohamed.slimane@univ-tours.fr</a>	Tuteur académique, Département Informatique
Jean-Yves RAMEL	<a href="mailto:jean-yves.ramel@univ-tours.fr">jean-yves.ramel@univ-tours.fr</a>	Tuteur académique, Département Informatique
Elena PIERAZZO	<a href="mailto:elena.pierazzo@univ-tours.fr">elena.pierazzo@univ-tours.fr</a>	Tuteur entreprise



# Avertissement

Ce document a été rédigé par Dylan MOTARD susnommé l'auteur.

L'entreprise Centre d'Études Supérieures de la Renaissance est représentée par Elena PIERAZZO susnommé le tuteur entreprise.

L'Ecole Polytechnique de l'Université de Tours est représentée par Mohamed SLIMANE et Jean-Yves RAMEL susnommés les tuteurs académiques.

Par l'utilisation de ce modèle de document, l'ensemble des intervenants du projet acceptent les conditions définies ci-après.

L'auteur reconnaît assumer l'entière responsabilité du contenu du document ainsi que toutes suites judiciaires qui pourraient en découler du fait du non respect des lois ou des droits d'auteur.

L'auteur atteste que les propos du document sont sincères et assume l'entière responsabilité de la véracité des propos.

L'auteur atteste ne pas s'approprier le travail d'autrui et que le document ne contient aucun plagiat.

L'auteur atteste que le document ne contient aucun propos diffamatoire ou condamnable devant la loi.

L'auteur reconnaît qu'il ne peut diffuser ce document en partie ou en intégralité sous quelque forme que ce soit sans l'accord préalable des tuteurs académiques et de l'entreprise.

L'auteur autorise l'école polytechnique de l'université François Rabelais de Tours à diffuser tout ou partie de ce document, sous quelque forme que ce soit, y compris après transformation en citant la source. Cette diffusion devra se faire gracieusement et être accompagnée du présent avertissement.



## Pour citer ce document

Dylan MOTARD, *CESR-DI : outil en ligne pour le balisage stand-off d'un appareil critique* : ,  
Projet Recherche & Développement, Ecole Polytechnique de l'Université de Tours, Tours,  
France, 2020-2021.

```
@mastersthesis{  
  author={MOTARD, Dylan},  
  title={CESR-DI : outil en ligne pour le balisage stand-off d'un appareil critique: },  
  type={Projet Recherche \& Développement},  
  school={Ecole Polytechnique de l'Université de Tours},  
  address={Tours, France},  
  year={2020-2021}  
}
```

# Table des matières

Liste des intervenants	<b>a</b>
Avertissement	<b>b</b>
Pour citer ce document	<b>c</b>
Table des matières	<b>i</b>
Table des figures	<b>v</b>
<b>1 Introduction</b>	<b>1</b>
1 Acteurs, enjeux et contexte .....	1
2 Objectifs .....	2
3 Bases méthodologiques.....	2
<b>2 Description générale</b>	<b>3</b>
1 Environnement du projet .....	3
2 Caractéristiques des utilisateurs .....	3
3 Fonctionnalités du système .....	3
4 Structure générale du système.....	4
<b>3 État de l'art</b>	<b>6</b>
1 L'apparat critique par la Text Encoding Initiative.....	6
1.1 La structure normale d'un fichier TEI.....	6
1.2 L'entrée des apparats critiques, les lectures et les témoins .....	7
1.3 Méthodes de liaison d'un appareil critique au texte .....	8
1.3.1 Méthode "location-referenced" .....	8
1.3.2 Méthode "double-end-point-attached" .....	9

1.3.3	Méthode "parallel segmentation" .....	10
2	Origine du projet coreBuilder.....	10
3	État de l'existant de l'application coreBuilder .....	11
3.1	Fonctionnement de l'application.....	11
3.1.1	Importation des fichiers XML .....	11
3.1.2	Définition des éléments stand-off.....	12
3.1.3	Création du balisage .....	13
3.1.4	Téléchargement du core .....	14
3.2	Critiques de l'application .....	14
<b>4</b>	<b>Analyse et conception</b> .....	<b>15</b>
1	Analyse .....	15
1.1	Reprise d'un travail déjà en cours .....	15
1.2	Ajout d'un lemme.....	16
1.3	Ajout du type de variation textuelle.....	16
1.4	Ajout d'une fenêtre de visualisation du core en cours .....	17
1.5	Ajout de formats de téléchargement du core.....	18
1.6	Changement des couleurs des types de variation textuelle .....	19
1.7	Ajout de différents types de visualisation des fichiers d'entrée .....	19
1.8	Répercussion de l'état du scrolling .....	20
2	Modélisation.....	20
2.1	Modèle .....	21
2.2	Vue .....	22
2.3	Collection .....	24
<b>5</b>	<b>Bilan et conclusion</b> .....	<b>25</b>
1	Bilan du semestre 9 .....	25
2	Planning du semestre 10 .....	25
	<b>Bibliographie</b> .....	<b>26</b>
	<b>Annexes</b> .....	<b>27</b>
<b>A</b>	<b>Cahier de spécifications</b> .....	<b>28</b>
1	Description des interfaces externes du logiciel.....	28
1.1	Interfaces homme-machine .....	28
1.2	Interfaces logiciel-logiciel.....	29
2	Spécifications fonctionnelles .....	29
2.1	Définition de la fonction de reprise d'un travail déjà en cours .....	29
2.1.1	Identification de la fonction .....	29

2.1.2	Description de la fonction .....	29
2.2	Définition de la fonction d'ajout d'un lemme.....	30
2.2.1	Identification de la fonction .....	30
2.2.2	Description de la fonction .....	31
2.3	Définition de la fonction d'ajout du type de variation textuelle .....	32
2.3.1	Identification de la fonction .....	32
2.3.2	Description de la fonction .....	32
2.4	Définition de la fonction d'ajout d'une fenêtre de visualisation du core en cours .....	33
2.4.1	Identification de la fonction .....	33
2.4.2	Description de la fonction .....	34
2.5	Définition de la fonction d'ajout de formats de téléchargement du core.....	34
2.5.1	Identification de la fonction .....	34
2.5.2	Description de la fonction .....	35
2.6	Définition de la fonction du changement des couleurs des types de variation textuelle .....	35
2.6.1	Identification de la fonction .....	35
2.6.2	Description de la fonction .....	36
2.7	Définition de la fonction d'ajout de différents types de visualisation des fichiers d'entrée.....	37
2.7.1	Identification de la fonction .....	37
2.7.2	Description de la fonction .....	37
2.8	Définition de la fonction de répercussion de l'état du scrolling.....	38
2.8.1	Identification de la fonction .....	38
2.8.2	Description de la fonction .....	38
3	Spécifications non fonctionnelles .....	39
3.1	Contraintes de développement et conception .....	39
3.2	Contraintes de fonctionnement et d'exploitation.....	40
3.2.1	Performances.....	40
3.2.2	Sécurité.....	40
<b>B</b>	<b>Cahier du développeur</b> .....	<b>41</b>
1	Introduction .....	41
2	Conventions de développement.....	41
3	Diagramme de classes du système .....	41
4	Description détaillée des classes .....	42
	La partie des modèles comporte plusieurs classes. ....	43
	La partie des vues comporte plusieurs classes. ....	43
	La partie des collections comporte plusieurs classes. ....	47

<b>C</b>	<b>Gestion de projet</b>	<b>49</b>
1	Planification .....	49
2	Méthodologie .....	49
3	Description des tâches.....	50
	Tâche 1 : compréhension du sujet .....	50
	Tâche 2 : expression des besoins .....	50
	Tâche 3 : rédaction de l'état de l'art.....	50
	Tâche 4 : rédaction du cahier de spécifications.....	50
	Tâche 5 : rédaction de l'analyse et du cahier du développeur .....	50
	Tâche 6 : rédaction du rapport du semestre 9 .....	50
	Tâche 7 : préparation à la soutenance du semestre 9.....	50
	Tâche 8 : développement de la fonctionnalité de reprise d'un travail déjà en cours.....	51
	Tâche 9 : développement de la fonctionnalité d'ajout d'un lemme ....	51
	Tâche 10 : développement de la fonctionnalité d'ajout du type de variation textuelle .....	51
	Tâche 11 : développement de la fonctionnalité d'ajout d'une fenêtre de visualisation du core en cours .....	51
	Tâche 12 : développement de la fonctionnalité d'ajout de formats de téléchargement du core.....	51
	Tâche 13 : développement de la fonctionnalité de changement des couleurs des types de variation textuelle.....	51
	Tâche 14 : développement de la fonctionnalité d'ajout de différents types de visualisation des fichiers d'entrée .....	51
	Tâche 15 : développement de la fonctionnalité facultative de réper- cussion de l'état du scrolling .....	52
	Tâche 16 : phase de tests.....	52
	Tâche 17 : rédaction du rapport du semestre 10.....	52
	Tâche 18 : préparation à la soutenance du semestre 10 .....	52
4	Gestion des risques .....	52



# Table des figures

## 2 Description générale

2.1	Fonctionnalités du système .....	4
2.2	Diagramme de classes de la structure générale du système .....	5

## 3 État de l'art

3.1	Fichier TEI.....	6
3.2	Leçons d'une entrée d'apparat critique .....	7
3.3	Lemme dans une entrée d'apparat critique .....	7
3.4	Attributs caractérisant les leçons.....	7
3.5	Groupement de leçons.....	8
3.6	Informations sur des témoins .....	8
3.7	Pointage vers une ancre .....	8
3.8	En-tête d'un fichier TEI avec la méthode "location-referenced" .....	9
3.9	Corps d'un fichier TEI avec la méthode "location-referenced" .....	9
3.10	Fichier TEI avec la méthode "double-end-point-attached" .....	9
3.11	Fichier TEI avec la méthode "parallel segmentation" .....	10
3.12	Duplication des leçons.....	11
3.13	Séparation des fichiers.....	11
3.14	Bouton d'ajout de fichiers .....	11
3.15	Zone d'import des fichiers .....	12
3.16	Fenêtre de définition des éléments stand-off.....	12
3.17	Ajout d'un élément au core .....	13
3.18	Élément ajouté au core .....	13
3.19	Groupement d'éléments .....	13
3.20	Visualisation du core .....	14

**4 Analyse et conception**

4.1	Mock-up de la zone d'importation des fichiers .....	15
4.2	Mock-up du choix du type de variation textuelle .....	16
4.3	Mock-up du menu de l'application .....	19
4.4	Diagramme de classes simplifié de l'application .....	21
4.5	Diagramme de classes du modèle de l'application .....	22
4.6	Diagramme de classes de la vue de l'application .....	23
4.7	Diagramme de classes de la collection de l'application .....	24

**A Cahier de spécifications**

A.1	Importation d'un mauvais format de fichier.....	28
A.2	Menu de l'application .....	29
A.3	Diagramme décrivant la fonction de reprise d'un travail déjà en cours .....	30
A.4	Diagramme décrivant la fonction d'ajout d'un lemme.....	31
A.5	Diagramme décrivant la fonction d'ajout du type de variation textuelle.....	33
A.6	Diagramme décrivant la fonction d'ajout d'une fenêtre de visualisation du core en cours .....	34
A.7	Diagramme décrivant la fonction d'ajout de formats de téléchargement du core.....	35
A.8	Diagramme décrivant la fonction du changement des couleurs des types de variation textuelle .....	36
A.9	Diagramme décrivant la fonction d'ajout de différents types de visualisation des fichiers d'entrée.....	38
A.10	Diagramme décrivant la fonction de répercussion de l'état du scrolling .....	39

**B Cahier du développeur**

B.1	Diagramme de classes du système .....	42
-----	---------------------------------------	----

**C Gestion de projet**

C.1	Planning du semestre 9 .....	49
C.2	Planning du semestre 10 .....	49

# 1

## Introduction

Cette première partie d'introduction présente le contexte de la réalisation du cahier de spécifications, concernant le projet de recherche et de développement "Outil en ligne pour le balisage stand-off d'un appareil critique".

### 1 Acteurs, enjeux et contexte

Dans ce projet, les acteurs mis en jeu sont la cliente Mme Elena Pierazzo, professeure en humanités numériques au Centre d'Études Supérieures de la Renaissance de l'Université de Tours, la maîtrise d'ouvrage représentée par MM. Mohamed Slimane et Jean-Yves Ramel, enseignants chercheurs au Département Informatique de l'École Polytechnique de l'Université de Tours et au Laboratoire d'Informatique Fondamentale et Appliquée de Tours, et la maîtrise d'œuvre représentée par Dylan Motard, étudiant en dernière année du cycle ingénieur en informatique de l'École Polytechnique de l'Université de Tours en collaboration avec Manon Ovide, étudiante en master 2 en Intelligence des Données de la Culture et des Patrimoines au Centre d'Études Supérieures de la Renaissance de l'Université de Tours.

Le projet s'inscrit dans le cadre d'une collaboration entre le Centre d'Études Supérieures de la Renaissance de l'Université de Tours et le Département Informatique de l'École Polytechnique de l'Université de Tours dont les enjeux sont d'assister le travail du chercheur en humanités numériques pour la transcription de documents anciens, et plus précisément un article sur l'écrivain Giangiorgio Trissino dont le texte constitue le cas d'étude.

Le contexte du projet est de pouvoir réaliser un balisage XML-TEI (Text Encoding Initiative) qui est le standard de données pour les textes anciens en lieu et place d'un balisage traditionnel à la main pouvant rapidement devenir difficile dans certains cas complexes. C'est dans cette optique que la poursuite du développement de l'outil Web pour le balisage en ligne de documents anciens dans le cadre des humanités numériques s'inscrit. Actuellement, cette application Web permet d'importer des fichiers XML encodés selon la Text Encoding Initiative qui sont issus de différents documents textuels anciens. Ces différents textes sont quasiment identiques, à l'exception de certains groupes de mots qui présentent des variations textuelles. À partir de ces différents fichiers, il est possible de créer un nouveau fichier XML que l'on appelle un core contenant un nouveau balisage XML qui va mettre en évidence les variations textuelles entre les différentes versions, dans le but de retenir uniquement les morceaux de texte les plus appropriés. Ce core est un ensemble de balises vides indiquant uniquement le lien entre les mots qui peut ensuite être utilisé

dans un appareil critique pour justifier les choix faits par l'utilisateur entre les textes de base qu'il a comparés pour établir le texte de son édition. Celui-ci peut également être mis en page. Ce projet franco-italien est également en connexion avec le Maryland Institute for Technology in the Humanities aux États-Unis qui est à l'origine du développement de l'application existante.

## 2 Objectifs

Les objectifs de ce projet sont de reprendre l'application Web en y ajoutant des fonctionnalités, d'améliorer l'interface utilisateur, de rajouter un aperçu graphique dynamique et de produire des fichiers de sortie aux formats XML, HTML et PDF. Les technologies JavaScript pour l'élaboration des différentes fonctionnalités de l'application, Sass pour la mise en forme graphique et XSLT pour la conversion des fichiers XML vers d'autres formats seront amenées à être utilisées.

## 3 Bases méthodologiques

Pour mener à bien le projet, la méthode de gestion de projets du cycle en V ainsi que le diagramme de Gantt pour suivre la planification des tâches du projet sont utilisés. Les outils Sonar pour la qualité du code et Git pour la gestion de versions permettant de suivre l'évolution de l'intégration des nouvelles fonctionnalités seront également utilisés. Concernant les règles de programmation, les noms de classes, méthodes, fonctions et variables devront être nommés de façon simple et précise et en anglais.

# 2

## Description générale

### 1 Environnement du projet

Ce projet s'inscrit dans la poursuite du développement d'une application Web existante. Concernant l'environnement logiciel, celle-ci se compose actuellement de Node.js, qui est un environnement d'exécution JavaScript permettant de compiler et d'exécuter le code JavaScript de l'application, ainsi que de gulp.js permettant d'automatiser certaines tâches. La poursuite du développement sera réalisée avec ces technologies sous le système d'exploitation Windows avec l'éditeur de code Visual Studio Code. Le projet ne dépend d'aucun environnement matériel.

### 2 Caractéristiques des utilisateurs

Le système sera amené à être pris en main par un seul type d'utilisateurs, à savoir les chercheurs en humanités numériques. Les utilisateurs possèdent des bases en informatique et notamment sur le langage XML afin de traiter le nouveau balisage XML dû aux variations textuelles dans l'interface utilisateur. Ils ont également une expérience de l'application puisque c'est une application existante et déjà déployée. Ces utilisateurs seront réguliers puisque cette application a pour but de simplifier leur travail et aucun droit d'accès n'est nécessaire puisque l'application est distribuée en open source et est donc complètement libre de droits.

### 3 Fonctionnalités du système

Les fonctions utilisateurs du système peuvent être représentées à l'aide d'un diagramme de cas d'utilisation.

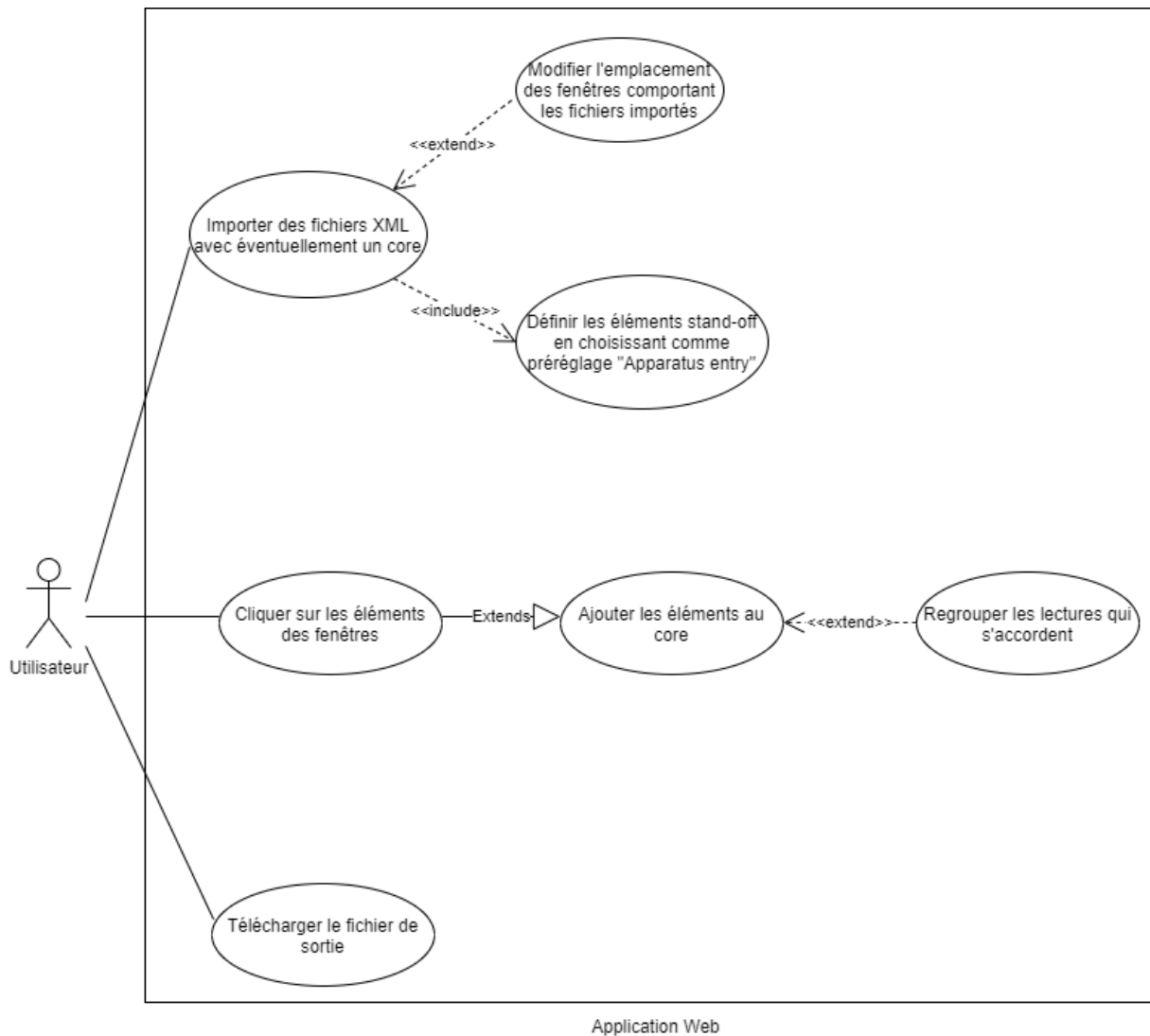
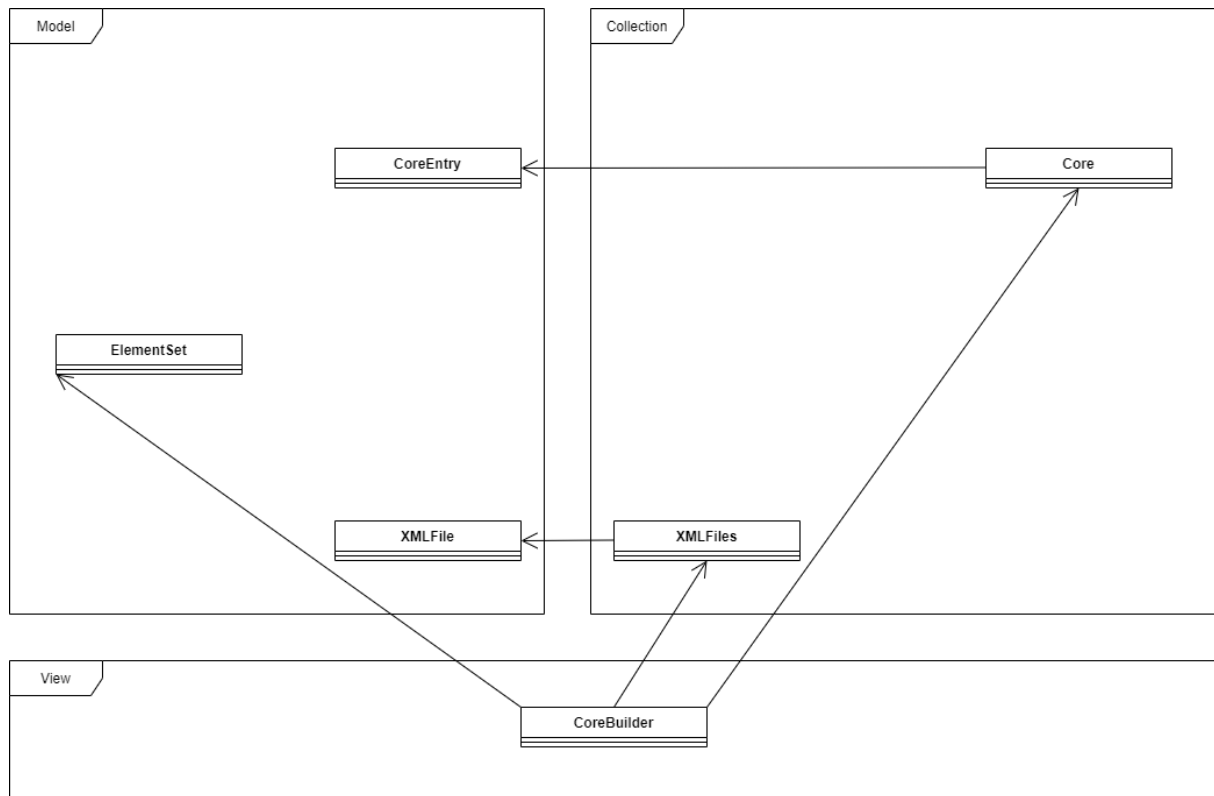


Figure 2.1 – Fonctionnalités du système

L'utilisateur a la possibilité d'importer des fichiers XML afin de pouvoir par la suite cliquer sur les éléments des fenêtres contenant ces fichiers et ainsi d'ajouter les éléments au fichier de sortie, appelé core. Au préalable, il doit définir le préréglage sur "Apparatus entry" pour avoir un système de balisage correspondant aux appareils critiques. Il peut également modifier l'emplacement des fenêtres contenant les fichiers et grouper les éléments de texte qui s'accordent, c'est-à-dire ceux dont la correspondance est la plus proche. Il peut enfin télécharger le core final qui constitue le fichier de sortie avec les balises créées vides. À noter que celui-ci a également la possibilité lors de l'importation, d'ajouter, en plus des fichiers XML d'entrée, un core afin de lui permettre de reprendre son travail.

## 4 Structure générale du système

La structure principale du système actuel se compose de trois parties, une pour les modèles, une pour les vues et une autre pour les collections. Ce type d'architecture est propre au framework JavaScript Backbone.js utilisé.



**Figure 2.2** – *Diagramme de classes de la structure générale du système*

Ce diagramme de classes très simplifié comporte uniquement les classes principales sans les attributs ni les méthodes et montre la structure interne générale de l'application et les différentes relations entre les trois parties. Le diagramme de classes complet et détaillé se situe à la figure [B.1](#) (Annexe B). La partie concernant les modèles contient les données des différents objets, celle relative aux collections représente des listes comportant les modèles et la partie concernant les vues permet la gestion des interactions avec l'utilisateur à partir des collections et des modèles.

# 3

## État de l'art

### 1 L'apparat critique par la Text Encoding Initiative

La Text Encoding Initiative permet l'encodage de documents anciens tels que les apparats critiques selon un système de balisage XML afin de mettre en évidence les variations textuelles pouvant exister d'un document à l'autre. [12 **Critical Apparatus**] [1]

#### 1.1 La structure normale d'un fichier TEI

Un fichier TEI de base se compose selon un balisage particulier.

```
<TEI xmlns="http://www.tei-c.org/ns/1.0">
  <teiHeader xml:lang="en">
    <!-- ... -->
  </teiHeader>
  <text xml:lang="fr">
    <body>
      <div>
        <!-- chapter one is in French -->
      </div>
      <div xml:lang="de">
        <!-- chapter two is in German -->
      </div>
      <div>
        <!-- chapter three is French -->
      </div>
    </body>
  </text>
</TEI>
```

Figure 3.1 – Fichier TEI

Dans ce fichier de base, la balise "TEI" indiquée délimite le fichier. La balise "teiHeader" représente les métadonnées et liste les témoins du texte, c'est-à-dire par exemple des manuscrits d'auteur, des éditions imprimées de l'œuvre, des traductions anciennes ou des citations d'une œuvre dans d'autres textes. La balise "text" permet de définir l'encodage du texte auquel des références sont faites aux témoins du texte présents dans les métadonnées.



## 1.2 L'entrée des appareils critiques, les lectures et les témoins

L'entrée des appareils critiques représente l'ensemble des variations textuelles entre plusieurs témoins d'un texte.

Concrètement, l'entrée des appareils critiques est encodée avec l'élément "app", qui contient également au moins une leçon qui est l'interprétation des caractères des mots et éventuellement un lemme décrivant la leçon et qui est la transcription admise. Différents attributs sont amenés à être utilisés, à savoir l'attribut "type" classifiant la variation, les attributs "from" et "to" identifiant le début et la fin du lemme et l'attribut "loc" indiquant la localisation de la variation.

Chaque entrée d'apparat critique contient au moins une leçon représentée par la balise "rdg".

```
<app>
  <rdg wit="#E1">Experience though noon Auctoritee</rdg>
  <rdg wit="#La">Experiment thoh noon Auctoritee</rdg>
  <rdg wit="#Ra2">Eryment though none auctorite</rdg>
</app>
```

Figure 3.2 – Leçons d'une entrée d'apparat critique

Cet exemple montre plusieurs leçons où l'on peut voir trois séquences de mots qui sont assez proches, extraites de trois témoins différents.

Afin de spécifier les lectures, les éléments "lem" et "rdg" représentant respectivement le lemme et la leçon sont utilisés.

```
<app>
  <lem wit="#E1 #Hg">Experience</lem>
  <rdg wit="#La" type="substantive">Experiment</rdg>
  <rdg wit="#Ra2" type="substantive">Eryment</rdg>
</app>
```

Figure 3.3 – Lemme dans une entrée d'apparat critique

À l'intérieur de ces deux éléments, il est possible d'indiquer des attributs comme les attributs "wit" qui contiennent un ou plusieurs pointeurs désignant les témoins et "type".

```
<app>
  <rdg wit="#La" varSeq="1">Experiment</rdg>
  <rdg wit="#Ra2" cause="abbreviation_loss"
    varSeq="2">Eryment</rdg>
</app>
```

Figure 3.4 – Attributs caractérisant les leçons

D'autres attributs peuvent être présents également tels que "cause" indiquant la cause d'une variation de leçon et "varSeq" indiquant la position d'une variation dans une séquence.

Il est possible d'améliorer la visibilité des lectures lorsque celles-ci ont des relations en commun par des éléments "rdgGrp" permettant de réaliser des groupements de leçons.

Ici, les deux leçons diffèrent à cause de l'orthographe, c'est pour cela qu'un groupement a été rendu possible selon le type orthographique.

```
<app>
  <lem wit="#E1 #Ra2">though</lem>
  <rdgGrp type="orthographic">
    <rdg wit="#La">thogh</rdg>
    <rdg wit="#Hg">thouh</rdg>
  </rdgGrp>
</app>
```

Figure 3.5 – Groupement de leçons

Dans une perspective d'amélioration du niveau de détails des témoins, il est possible d'ajouter des balises "witDetail" permettant d'avoir des renseignements supplémentaires sur les témoins.

```
<app type="substantive">
  <lem wit="#E1 #Hg">Experience</lem>
  <witDetail wit="#E1">Ornamental capital.</witDetail>
  <rdg wit="#Ha4">Experiens</rdg>
</app>
```

Figure 3.6 – Informations sur des témoins

Dans ce cas, le contenu de l'élément "witDetail", par correspondance du témoin, apporte des informations additionnelles sur le lemme.

```
<app>
  <lem>Nondum</lem>
  <rdg wit="#G #P" xml:id="rdg1.1nundum"
    ana="#orthographical">nundum</rdg>
  <witDetail wit="#G" target="#rdg1.1nundum">corr. <ref target="#G1">G<hi rend="super">1</hi>
    </ref>
  </witDetail>
</app>
```

Figure 3.7 – Pointage vers une ancre

L'ajout de l'attribut "target" au sein de la balise "ref" permet de pointer vers l'ancre en question.

### 1.3 Méthodes de liaison d'un appareil critique au texte

Il est tout d'abord important d'avoir conscience qu'il existe deux différents types d'appareils critiques, ceux en ligne signifiant qu'ils sont intérieurs aux textes et ceux qui sont externes, c'est-à-dire extérieurs aux textes correspondant au terme "stand-off".

Les méthodes spécifiées ci-après sont des méthodes d'encodage des variations textuelles.

#### 1.3.1 Méthode "location-referenced"

Cette méthode permet de lier l'apparat critique au texte de base en indiquant uniquement le bloc de texte sur lequel une variation existe par un schéma de référence canonique ou par un numéro de ligne dans l'édition.

```
<variantEncoding method="location-referenced"
  location="external"/>
```

Figure 3.8 – En-tête d'un fichier TEI avec la méthode "location-referenced"

Pour pouvoir utiliser une des méthodes, il est nécessaire d'indiquer son nom ainsi que si l'apparat critique est intérieur ou extérieur au texte.

```
<text>
  <body>
    <div n="WBP" type="prologue">
      <head>The Prologue of the Wyves Tale of Bathe</head>
      <l n="1">Experience though noon Auctoritee</l>
      <l>Were in this world ...</l>
    </div>
  </body>
</text>
```

Figure 3.9 – Corps d'un fichier TEI avec la méthode "location-referenced"

Pour chacun des blocs de texte dans lesquels une variation est détectée, une balise "app" est créée avec un attribut "loc" représentant la position, comme un numéro de ligne, de vers ou de paragraphe par exemple, de la variation dans le texte de base. Cet attribut a pour valeur les valeurs des attributs "n" séparés par des espaces. À l'intérieur, les balises "rdg" représentent les différentes interprétations de lecture du texte de base. Une balise "lem" est également présente afin de préciser la portion du texte concernée par les erreurs de lecture.

C'est une méthode qui est relativement similaire à ce qui peut être fait avec les éditions papier.

### 1.3.2 Méthode "double-end-point-attached"

Cette méthode permet d'avoir une détection de variations plus précise que la méthode précédente étant donné que le début et la fin du lemme permettent d'avoir une précision accrue comparée à la méthode précédente. Afin de délimiter le lemme, les attributs "from" et "to" sont spécifiés dans les balises "app" pour indiquer le début et la fin du lemme. Dans le cas de l'absence de ces attributs, la balise "anchor" qui est un point d'ancrage spécifiant un identifiant à un endroit du texte doit être utilisée afin de pallier cela.

```
<variantEncoding method="double-end-point"
  location="internal"/>
<!-- ... -->
<l n="1" xml:id="wbp.1">Experience
  <app from="#wbp.1">
    <rdg wit="#La">Experiment</rdg>
    <rdg wit="#Ra2">Eryment</rdg>
  </app>
  though noon Auctoritee</l>
<l>Were in this world ...</l>
```

Figure 3.10 – Fichier TEI avec la méthode "double-end-point-attached"

À noter que comme pour la première méthode, l'apparat critique peut être intérieur ou extérieur au texte de base.

Cette méthode présente les avantages de gérer les chevauchements, de permettre ainsi également un encodage collaboratif et d'avoir une très bonne précision. L'inconvénient est que celle-ci n'est pas très pratique à utiliser puisqu'elle est assez lourde.

### 1.3.3 Méthode "parallel segmentation"

Avec cette méthode, contrairement à la seconde, aucune des deux variations ne peut se chevaucher. Les textes comparés sont divisés en segments permettant de réaliser une comparaison directe d'un morceau de texte d'un témoin dans un autre témoin.

Chaque segment de texte présentant une variation marque l'apparition d'une balise "app".

```
<variantEncoding method="parallel-segmentation"
  location="internal"/>
<!-- ... -->
<l n="1">
  <app>
    <lem wit="#E1 #Hg">Experience</lem>
    <rdg wit="#La">Experiment</rdg>
    <rdg wit="#Ra2">Eryment</rdg>
  </app> though noon Auctoritee
</l>
<l>Were in this world ...</l>
```

Figure 3.11 – Fichier TEI avec la méthode "parallel segmentation"

Cette méthode permet de gérer uniquement les appareils critiques intérieurs au texte, contrairement aux deux autres méthodes qui gèrent les deux types. Celle-ci reste néanmoins la plus utilisée puisqu'elle est relativement simple d'utilisation, malgré le fait qu'elle offre moins de possibilités par rapport à la seconde méthode.

## 2 Origine du projet coreBuilder

Cet outil Web a été conçu à la base afin de simplifier la création du balisage à distance, c'est-à-dire dans un fichier dédié à cela et non pas directement dans les fichiers d'entrée. Il s'est notamment inscrit dans le cadre de la complexité du balisage manuel pour l'édition du livret pour le projet numérique Freischütz. [Why TEI Stand-off Markup Authoring Needs Simplification] [2]

À la base, l'outil coreBuilder était destiné au domaine musical et non à celui des documents anciens. En effet, l'objectif de l'édition du livret pour le projet numérique Freischütz était de créer une édition numérique critique d'un opéra romantique et ainsi de transcrire notamment les partitions musicales manuscrites.

La méthode "parallel segmentation" vue précédemment ne pouvait pas être utilisée dans ce cas de figure puisque la transcription nécessitait d'avoir l'encodage des variations textuelles à l'extérieur du texte, ce que ne permet pas cette méthode. C'est pour cela que la méthode "double-end-point-attached" a été un temps envisagée puisqu'elle offre la possibilité d'effectuer un balisage à distance en dehors du texte, en spécifiant le début et la fin du lemme sur lequel s'appuyer afin de détecter les variations textuelles. Cependant, celle-ci n'est pas idéale puisqu'elle nécessite de dupliquer des lignes de code dans le cas où un morceau de texte soit souligné par exemple.

La solution retenue a donc été de créer un fichier séparé encodant les variations textuelles où les balises "rdg" contenant les groupes de mots ayant des variations ont des pointeurs dans l'encodage des sources.

```

<speaker>
  <app>
    <rdg wit="#W1">Agathe</rdg>
    <rdg wit="#W2"><hi rend="underline">Agathe</hi></rdg>
  </app>
</speaker>

```

Figure 3.12 – Duplication des leçons

Source KA-tx15.xml	Source A-pt.xml
<pre> &lt;l xml:id="KA-tx15_l1"&gt;Sie erquickte,&lt;/l&gt; &lt;l xml:id="KA-tx15_l2"&gt;Und &lt;w xml:id="KA-tx15_w1"&gt;bestricke&lt;/w&gt; &lt;/l&gt; &lt;l xml:id="KA-tx15_l3"&gt;Und beglücke,&lt;/l&gt; </pre>	<pre> &lt;l xml:id="A-pt_l1"&gt;Sie erquickte,&lt;/l&gt; &lt;l xml:id="A-pt_l2"&gt;und beglücke,&lt;/l&gt; &lt;l xml:id="A-pt_l3"&gt;und &lt;w xml:id="A-pt_w1"&gt;bestrikke.&lt;/w&gt; &lt;/l&gt; </pre>
<b>Apparatus file</b> <pre> &lt;app&gt;   &lt;rdg wit="#KA-tx15"&gt;     &lt;ptr target="KA-tx15.xml#KA-tx15_l2"/&gt;     &lt;ptr target="KA-tx15.xml#KA-tx15_l3"/&gt;   &lt;/rdg&gt;   &lt;rdg wit="#A-pt"&gt;     &lt;ptr target="A-pt.xml#A-pt_l2"/&gt;     &lt;ptr target="A-pt.xml#A-pt_l3"/&gt;   &lt;/rdg&gt; &lt;/app&gt; &lt;app&gt;   &lt;rdg wit="#KA-tx15"&gt;     &lt;ptr target="KA-tx15.xml#KA-tx15_w1"/&gt;   &lt;/rdg&gt;   &lt;rdg wit="#A-pt"&gt;     &lt;ptr target="A-pt.xml#A-pt_w1"/&gt;   &lt;/rdg&gt; &lt;/app&gt; </pre>	

Figure 3.13 – Séparation des fichiers

### 3 État de l'existant de l'application coreBuilder

#### 3.1 Fonctionnement de l'application

L'application Web "coreBuilder" actuelle a pour vocation de limiter les erreurs induites par la création manuelle d'un balisage stand-off, consistant à créer un balisage à distance à partir des fichiers XML d'entrée. [coreBuilder] [3]

##### 3.1.1 Importation des fichiers XML

L'application offre tout d'abord la possibilité d'importer des fichiers XML sur lesquels l'utilisateur pourra ensuite travailler par la suite.

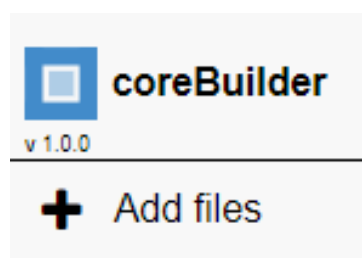
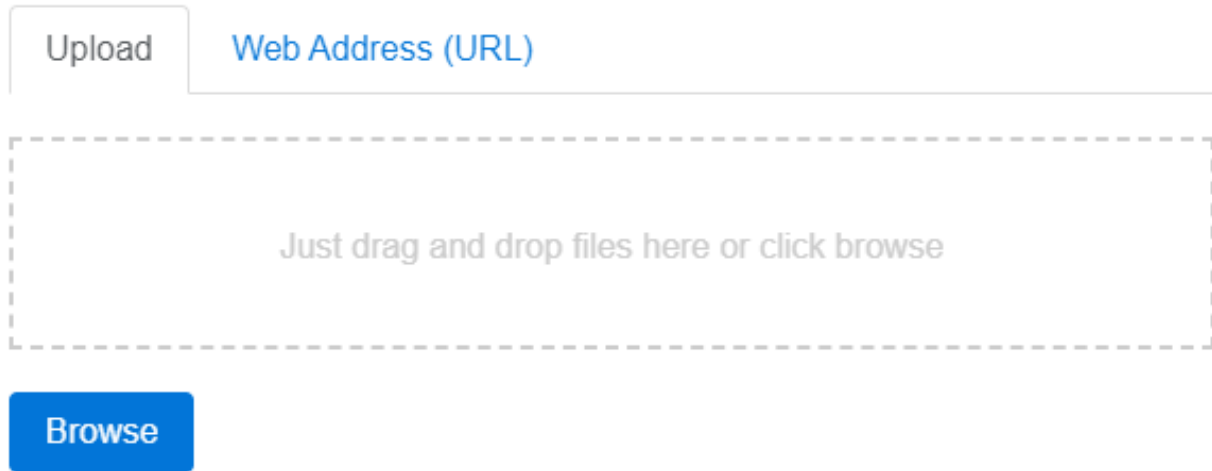


Figure 3.14 – Bouton d'ajout de fichiers

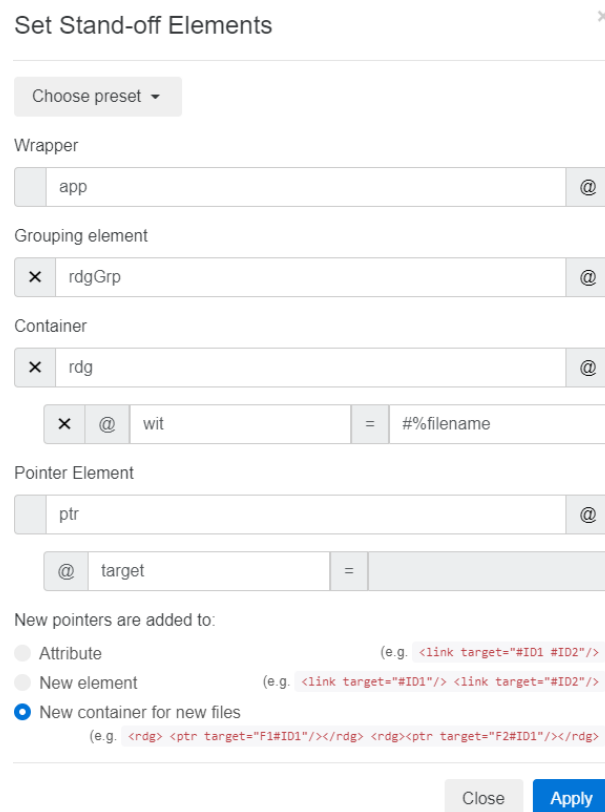
Pour cela, l'utilisateur peut cliquer sur le bouton "Add files" du menu, ce qui affiche par la suite un pop-up permettant le glisser-déposer des fichiers souhaités ou la saisie d'une URL pour une importation en ligne.



**Figure 3.15** – Zone d'import des fichiers

### 3.1.2 Définition des éléments stand-off

Il est ensuite possible de définir les éléments stand-off selon un préréglage à sélectionner. Tout d'abord, l'utilisateur doit cliquer sur le bouton "Stand-off" du menu précédent, ce qui engendre l'apparition d'un pop-up.



**Figure 3.16** – Fenêtre de définition des éléments stand-off

Ici, le préréglage "Apparatus entry (<app>, <rdg>)" a été choisi afin de disposer des différents éléments relatifs aux appareils critiques avec la Text Encoding Initiative.

### 3.1.3 Création du balisage

Une fois les fichiers chargés dans l'application, il est possible de procéder à la création du balisage en cliquant sur les identifiants des éléments "xml:id" qui sont différents d'un fichier à l'autre.



Figure 3.17 – Ajout d'un élément au core

Cette action permet d'ajouter l'élément au core.

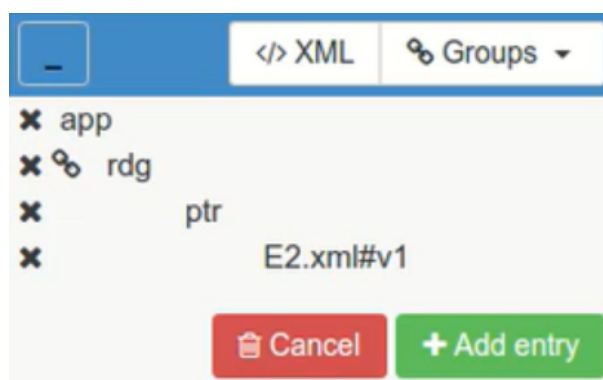


Figure 3.18 – Élément ajouté au core

Lorsque plusieurs éléments du même type sont sélectionnés, il est possible de grouper les éléments de texte qui se ressemblent et donc de laisser les autres à part. Pour réaliser cela, il suffit de cliquer sur "Groups", puis sur "New group" et enfin de cliquer sur les petites icônes permettant de faire le groupement.

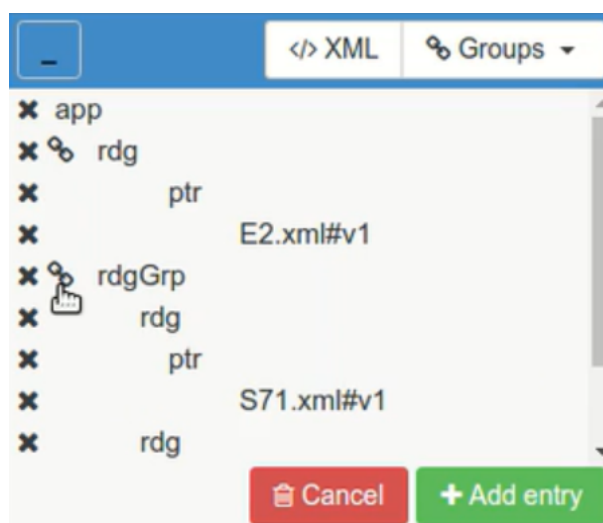


Figure 3.19 – Groupement d'éléments

### 3.1.4 Téléchargement du core

Pour la réalisation de cette dernière étape, l'utilisateur est amené à cliquer sur l'onglet "View Core" du menu et enfin à cliquer sur le bouton "Download Core" dans le pop-up qui s'affiche.



Figure 3.20 – Visualisation du core

## 3.2 Critiques de l'application

L'application développée fonctionne correctement et permet d'obtenir le résultat voulu. Néanmoins, cette dernière n'est pas très pratique d'un point de vue ergonomique et de l'utilisation. C'est pour cela que certains ajouts et certaines modifications peuvent être intéressants à intégrer.

Actuellement, lorsque le groupement est réalisé, aucun choix sur le type de variation qu'il soit orthographique, sémantique, de ponctuation, d'omission, de répétition ou autre n'est proposé à l'utilisateur. La ponctuation devrait donc être tokenisée afin de pouvoir la gérer. De plus, il est impossible de sélectionner un lemme sur lequel s'appuyer pour déterminer le type des variations. Il serait également intéressant d'avoir des couleurs spécifiques pour chacun des types de variation sur les éléments qui ont été sélectionnés afin de gagner en lisibilité. Ces couleurs pourraient aussi être changeables par le biais d'une palette de couleurs.

Dans le but d'améliorer la lisibilité des fenêtres, il pourrait être judicieux d'avoir la possibilité d'interchanger la vue de chacune des fenêtres issues des fichiers d'entrée, entre du code et du texte et également de répercuter l'état du scrolling de la fenêtre actuelle sur toutes les autres fenêtres qui correspondent aux fichiers d'entrée.

Pour mieux se rendre compte de l'évolution du texte en train d'être balisé, il faudrait prévoir l'ajout d'une fenêtre de visualisation du core en cours et faire en sorte que la modification et le téléchargement de celui-ci au format HTML et PDF, en plus du XML, soit possible, ce qui entraînera la suppression de l'onglet "View Core" qui n'aura plus d'utilité.

Enfin, le fait de ne pas pouvoir continuer son travail est un aspect très gênant, il faut envisager que l'utilisateur télécharge le core qu'il voudra reprendre par la suite et qu'il l'importe en tant que core, qui sera une option facultative, en plus des fichiers d'entrée lorsqu'il souhaite poursuivre son travail.



# 4

## Analyse et conception

### 1 Analyse

Cette partie traite de l'analyse dont des compléments d'information sont disponibles dans la partie des spécifications fonctionnelles 2(Annexe A).

#### 1.1 Reprise d'un travail déjà en cours

La réalisation de la tâche qui consiste en la reprise d'un travail déjà en cours doit permettre à l'utilisateur s'il souhaite poursuivre son travail, d'importer en plus des fichiers, le core qu'il a préalablement téléchargé avant d'avoir quitté l'application.

Actuellement, l'application ne permet d'importer que des fichiers d'entrée, ce qui oblige donc l'utilisateur à commencer un nouveau travail dès lors qu'il utilise l'application.

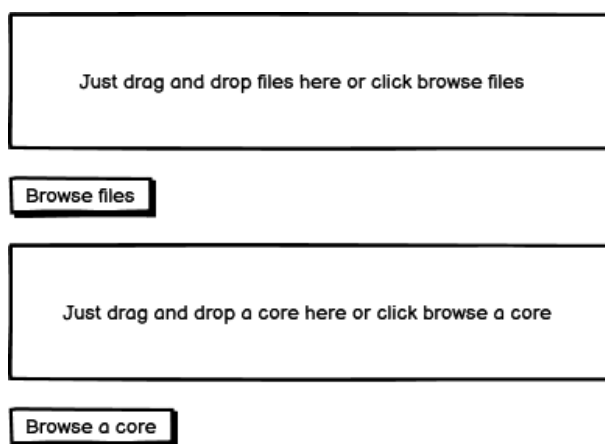


Figure 4.1 – Mock-up de la zone d'importation des fichiers

L'objectif est donc d'ajouter une seconde zone de dépôt, avec le bouton correspondant afin de donner la possibilité à l'utilisateur s'il le souhaite d'importer le core correspondant aux fichiers d'entrée.

Lorsque l'utilisateur prend la décision d'arrêter son travail pour le reprendre plus tard, il pourra cliquer sur un bouton qui sera ajouté sur la fenêtre de visualisation du core. Cette action téléchargera le core au format XML.

## 1.2 Ajout d'un lemme

La réalisation de la tâche consistant à l'ajout d'un lemme doit permettre à l'utilisateur d'ajouter des lemmes, s'il a défini qu'il souhaitait utiliser le lemme.

La possibilité pour l'utilisateur d'utiliser ou non le lemme sera définie lors de la phase d'importation des fichiers. Si l'utilisateur importe uniquement des fichiers d'entrée, il aura la possibilité de définir s'il souhaite utiliser le lemme ou non. En revanche, s'il importe également un core, l'application détectera automatiquement si ce dernier contient au moins une balise correspondant à un lemme et la possibilité ne sera donc pas offerte à l'utilisateur.

Dans le cas où l'utilisateur ait fait le choix d'utiliser le lemme, lorsqu'il sélectionne un groupe de mots dans un des fichiers importés qui n'a pas été sélectionné auparavant dans un des autres fichiers, celui-ci est directement ajouté en tant que lemme dans le core, sinon il est ajouté au core normalement. Dans le cas où l'utilisateur n'ait pas fait le choix d'utiliser le lemme, celui-ci est ajouté au core de façon classique.

## 1.3 Ajout du type de variation textuelle

La réalisation de la tâche qui consiste en l'ajout du type de variation textuelle permet à l'utilisateur, dans le cas où son projet ait été défini avec l'utilisation du lemme, de caractériser s'il le désire la variation textuelle d'un groupe de mots ajouté au core et pour lequel un lemme a déjà été spécifié. Lorsqu'il clique sur le groupe de mots en question, un pop-up s'affiche dans lequel il peut choisir une variation textuelle entre une variation orthographique, sémantique, de ponctuation, d'omission ou de répétition.

- ☐ Spelling variation
- ☐ Semantic variation
- ☐ Variation of punctuation
- ☐ Variation of omission
- ☐ Variation of repetition
- ☒ Other variation



Figure 4.2 – Mock-up du choix du type de variation textuelle

Si la variation textuelle qu'il souhaite apporter n'est pas présente dans le pop-up qui s'affiche, l'utilisateur a la possibilité d'en ajouter une autre en sélectionnant l'option "Other variation"

en renseignant le nom du type de la variation textuelle dans le champ prévu à cet effet qui est visible que lorsque cette option est sélectionnée.

Dans le cas où l'utilisateur reprenne un travail déjà en cours, l'application détectera automatiquement lors de la phase d'importation du core, les éventuels autres types de variation textuelle qui ne sont pas présents par défaut, à savoir orthographique, sémantique, de ponctuation, d'omission et de répétition, et ajoutera ces nouveaux types de variation textuelle dans la liste actuelle.

### 1.4 Ajout d'une fenêtre de visualisation du core en cours

La réalisation de la tâche qui consiste en l'ajout d'une fenêtre de visualisation du core en cours doit permettre à l'utilisateur de visualiser directement l'avancement de la totalité de core.

Actuellement, l'application permet de visualiser directement uniquement l'entrée d'apparat critique en cours. Pour pouvoir visualiser la totalité du core, il est nécessaire de se rendre dans l'onglet "View Core" du menu de l'application.

L'objectif est donc d'ajouter la totalité du core au niveau de cette fenêtre de visualisation où la visualisation au format XML sera toujours disponible, mais également d'ajouter un autre type de visualisation au format HTML par l'intermédiaire d'un fichier CSS.

```

1  .td-app {
2      vertical-align: top;
3      width: 170px;
4      border-left: solid 1px;
5      border-right: solid 1px;
6      padding-left: 10px
7  }
8
9  #appbox {
10     font-family: Verdana;
11     font-size: 8pt;
12     position: fixed;
13     width: 150px;
14 }
15
16 body {
17     padding-left: 80px;
18     padding-right: 80px;
19     text-align: justify;
20     font-size: 30px;
21     font-family: roman, 'times new roman', times, serif;
22     display: block;
23 }
24
25 table {
26     font-size: 12pt;
27     text-align: justify;
28     padding: 20px 20px 20px 20px
29 }
30
31 .td-text {
32     padding: 20px 20px 20px 20px;
33     width: 600px;
34 }

```

Ce fichier CSS permet donc de styliser le core qui est visualisé au format HTML.

## 1.5 Ajout de formats de téléchargement du core

La réalisation de la tâche d'ajout de formats de téléchargement pour le core consiste à donner à l'utilisateur la possibilité de télécharger le core aux formats HTML et PDF en plus du format XML.

Actuellement, l'application ne permet de télécharger le core qu'au format XML. Le but est donc d'ajouter un bouton de téléchargement au niveau de la fenêtre de visualisation du core et d'inclure une liste déroulante contenant les différents types de formats possibles, à savoir XML, HTML et PDF.

Pour réaliser cette tâche, la technologie XSLT est amenée à être utilisée. Celle-ci permet de transformer un fichier XML en différents autres formats de fichier tels que HTML et PDF. Cela permet donc de prendre les différentes balises XML et d'inclure des éléments "xsl" sur ces

différentes balises qui permettent de réaliser la conversion entre les formats. Pour ce qui concerne le format HTML, le fichier CSS précédemment évoqué sera utilisé pour le style du fichier HTML généré.

## 1.6 Changement des couleurs des types de variation textuelle

La réalisation de la tâche de changement des couleurs des types de variation textuelle permet à l'utilisateur de spécifier les couleurs qu'il souhaite définir pour chaque type de variation textuelle en sachant que par défaut la variation orthographique est en vert, la variation sémantique en rouge, la variation de ponctuation en orange, la variation d'omission en bleu et la variation de répétition en violet. Les éventuels autres types de variation textuelle se verront affecter une couleur automatiquement par l'application qui pourra ensuite être modifiée par l'utilisateur en cas de besoin.

Concrètement, un onglet "Colors variations" est ajouté au menu de l'application.



Figure 4.3 – Mock-up du menu de l'application

Le clic sur cet onglet ouvre un pop-up permettant de modifier chacune des couleurs des différents types de variation textuelle par l'intermédiaire d'une palette de couleurs.

## 1.7 Ajout de différents types de visualisation des fichiers d'entrée

La réalisation de la tâche d'ajout de différents types de visualisation des fichiers d'entrée consiste à donner à l'utilisateur la possibilité de permuter le mode de visualisation des fichiers d'entrée entre un affichage au format XML qui est le mode de visualisation par défaut et une visualisation textuelle en passant par le format HTML.

Concrètement, un bouton est ajouté en haut de chacune des fenêtres de visualisation comprenant les fichiers d'entrée. À chaque clic sur ce bouton, le mode de visualisation permute entre un

affichage au format XML et un autre au format HTML. La technologie XSLT est de nouveau utilisée afin de passer du format XML au format HTML. Pour gérer le style des fichiers HTML, les balises "rendition" présentes dans les balises "teiHeader" des fichiers XML originaux sont utilisées.

## 1.8 Répercussion de l'état du scrolling

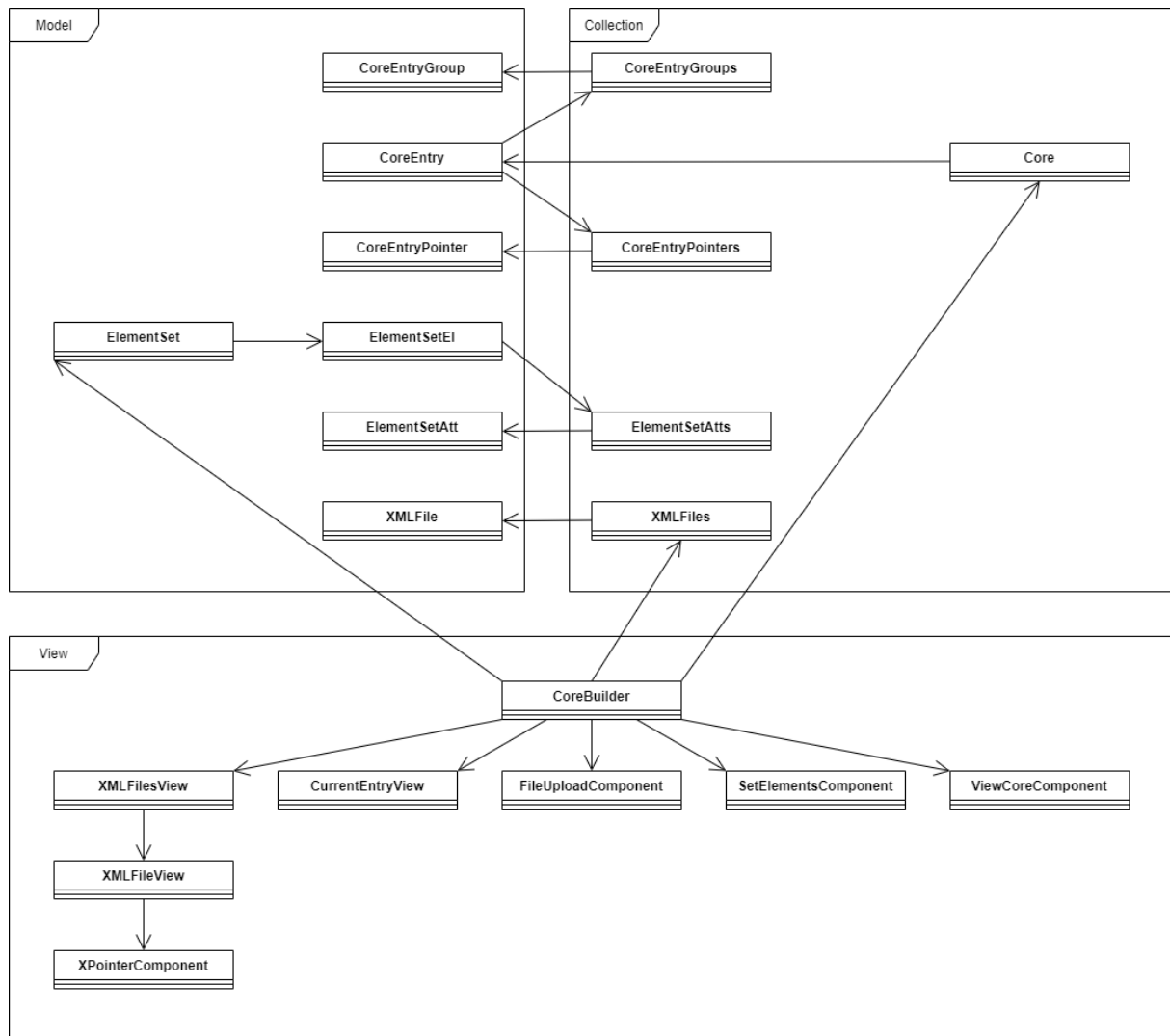
La réalisation de la tâche de répercussion de l'état du scrolling permet à l'utilisateur de gérer plus facilement la visualisation de chacune des fenêtres de visualisation des fichiers d'entrée étant donné que lorsqu'il scrolle dans une des fenêtres, l'état de scrolling de celle-ci est répercuté dans les autres.

Actuellement, lorsque l'utilisateur scrolle dans une fenêtre de visualisation, les autres fenêtres restent statiques, ce qui n'est pas très pratique pour pouvoir comparer les groupes de mots qu'il analyse étant donné que ces derniers se situent au même niveau dans chacune des fenêtres.

Lorsque l'utilisateur scrolle dans une fenêtre de visualisation d'un fichier d'entrée, l'application détecte le nombre de pixels qui sépare l'ascenseur du haut de la fenêtre. Ainsi, les ascenseurs des autres fenêtres sont ajustés automatiquement de la même manière. Cela a pour effet d'avoir le même état de scrolling dans chacune des fenêtres en permanence.

## 2 Modélisation

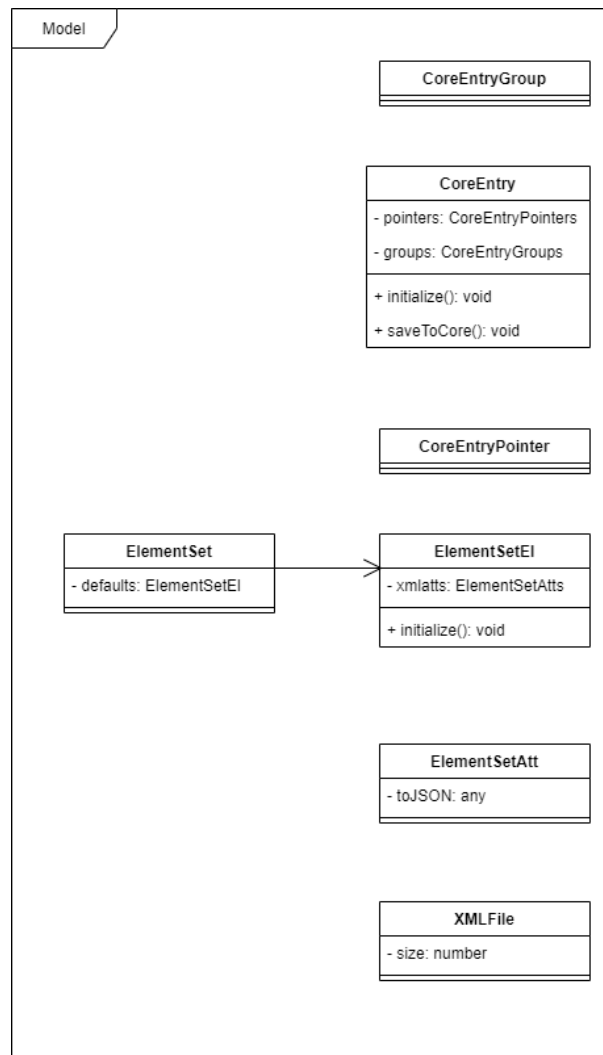
L'architecture de l'application est réalisée à l'aide du framework JavaScript Backbone.js selon un patron de conception MVC (Model View Collection). Les modèles contiennent les données des objets, les vues permettent de gérer des parties de code HTML et les collections contiennent des listes d'objets. L'utilisation de ce framework se justifie par le fait que l'application comporte une seule page principale avec beaucoup d'interactions avec l'utilisateur gérées par des événements. Des compléments d'information sont disponibles dans l'annexe [B](#).



**Figure 4.4** – Diagramme de classes simplifié de l'application

Ce diagramme de classes est un diagramme simplifié sans les attributs ni les méthodes. Chacune des classes hérite de Backbone.Model, Backbone.View ou Backbone.Collection afin de bénéficier des méthodes du framework Backbone.js.

## 2.1 Modèle



**Figure 4.5** – Diagramme de classes du modèle de l'application

Ce diagramme de classes représente le modèle de l'application.

- La classe "CoreEntryGroup" représente un groupement effectué dans le core.
- La classe "CoreEntry" représente une entrée d'apparat critique du core.
- La classe "CoreEntryPoint" représente une expression créée dans le core par la sélection de texte directement dans un fichier XML.
- La classe "ElementSet" représente les éléments de balisage stand-off par défaut.
- La classe "ElementSetEl" représente les attributs XML en fonction du type de balisage stand-off sélectionné par l'utilisateur.
- La classe "ElementSetAtt" représente l'attribut XML sélectionné par l'utilisateur dans un fichier XML.
- La classe "XMLFile" représente un fichier XML.



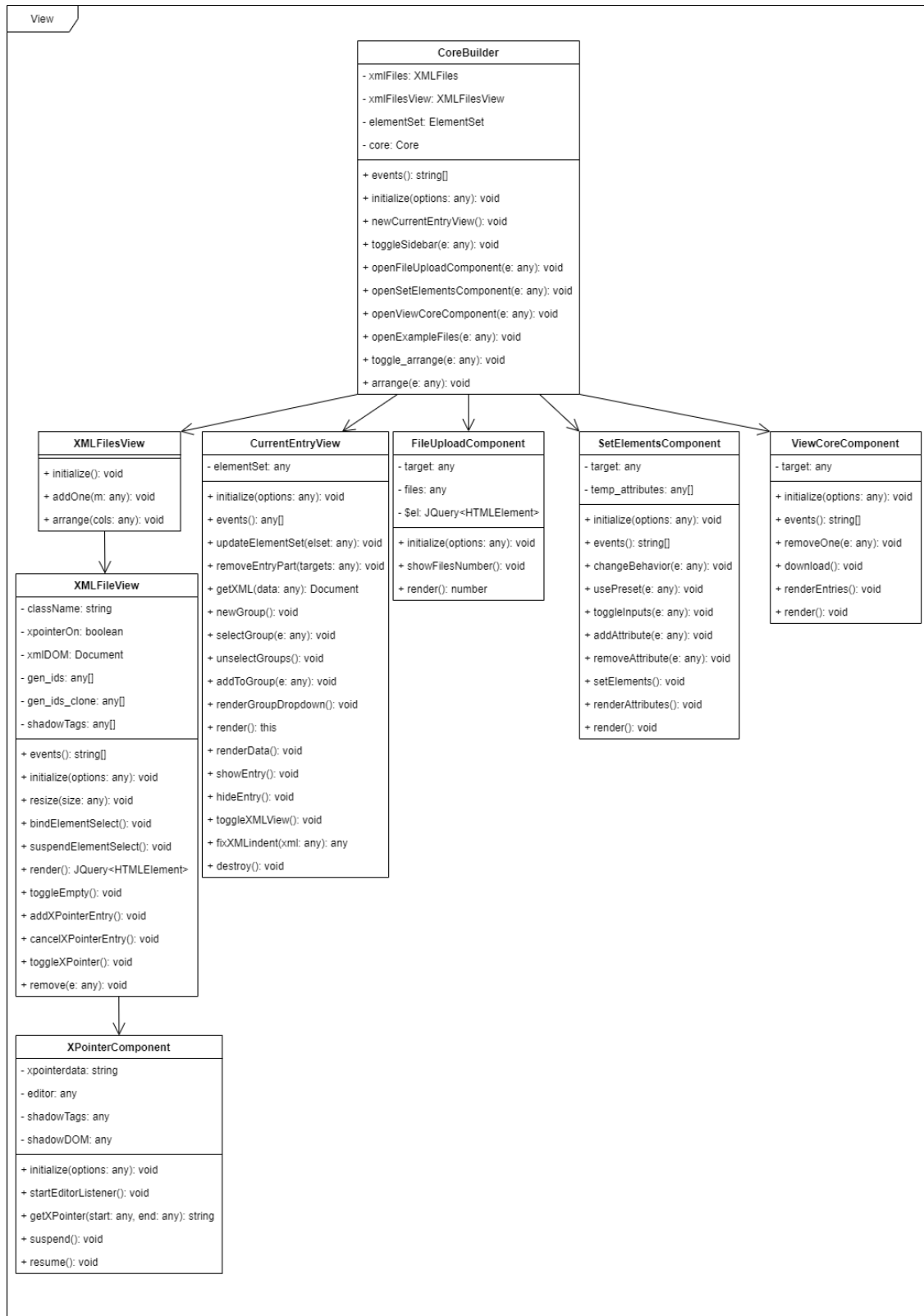


Figure 4.6 – Diagramme de classes de la vue de l'application

Ce diagramme de classes représente la vue de l'application.

- La classe "CoreBuilder" gère les interactions principales avec l'utilisateur de l'application.
- La classe "XMLFilesView" gère les interactions de l'ensemble des fichiers XML.
- La classe "XMLFileView" gère les interactions spécifiques à chacun des fichiers XML. La classe "XPointerComponent" gère les interactions concernant la création d'expressions dans

- les fichiers XML d'entrée par l'utilisateur.
- La classe "CurrentEntryView" gère les interactions avec la fenêtre de l'entrée d'apparat critique en cours.
- La classe "FileUploadComponent" gère les interactions lors de l'importation des fichiers.
- La classe "SetElementsComponent" gère les interactions lors de la définition des éléments de balisage stand-off.
- La classe "ViewCoreComponent" gère les interactions du pop-up de visualisation du core.

### 2.3 Collection

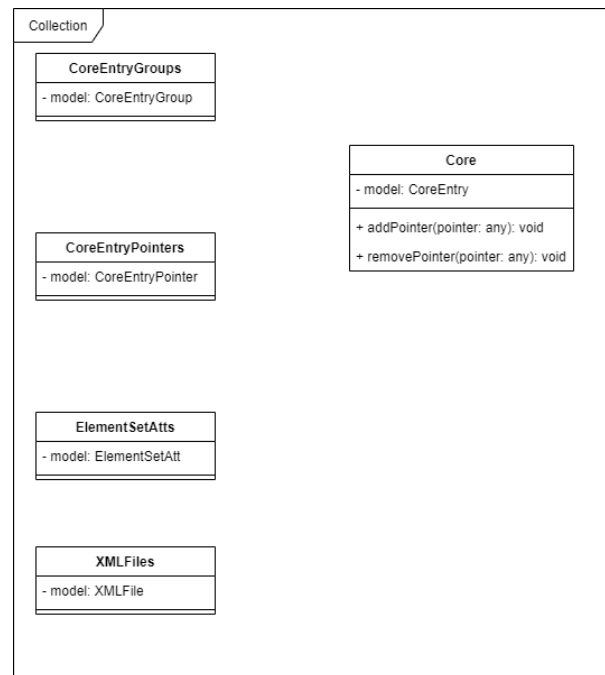


Figure 4.7 – Diagramme de classes de la collection de l'application

Ce diagramme de classes représente la collection de l'application.

- La classe "CoreEntryGroups" regroupe les modèles "CoreEntryGroup" et représente donc l'ensemble des groupements effectués dans le core.
- La classe "Core" regroupe les modèles "CoreEntry" et représente donc l'ensemble des entrées d'apparat critique du core.
- La classe "CoreEntryPointers" regroupe les modèles "CoreEntryPointer" et représente donc l'ensemble des expressions créées dans le core par la sélection de texte directement dans les fichiers XML.
- La classe "ElementSetAtts" regroupe les modèles "ElementSetAtt" et représente donc l'ensemble des attributs XML sélectionnés par l'utilisateur dans les fichiers XML.
- La classe "XMLFiles" regroupe les modèles "XMLFile" et représente donc l'ensemble des fichiers XML.

# 5

## Bilan et conclusion

### 1 Bilan du semestre 9

Au cours du semestre 9, j'ai été amené à réaliser différentes tâches afin de mener à bien la partie concernant la recherche du projet conformément au planning de la figure C.1(Annexe C).

Les tâches prévues au cours de ce semestre consistaient en la compréhension du sujet, l'expression des besoins, la rédaction de l'état de l'art, des spécifications, du cahier du développeur et du rapport, ainsi qu'en la préparation à la soutenance.

Toutes les tâches ont été réalisées en respectant les délais fixés par le planning précédent. Aucun retard n'est donc à déplorer et il n'y a donc pas de reste à faire pour la partie de la recherche issue de ce semestre. Cela permettra donc d'aborder la partie du développement sereinement, et ce dès le début du mois de janvier.

### 2 Planning du semestre 10

Pour le semestre 10, plusieurs tâches concernant le développement du projet ont été planifiées conformément au planning de la figure C.2(Annexe C).

Le développement commencera au début du mois de janvier en commençant par les tâches les plus prioritaires. Ainsi, le développement des fonctionnalités primordiales sera réalisé en premier, puis celui des fonctionnalités secondaires et enfin celui de la fonctionnalité facultative. Une phase de tests fonctionnels et unitaires sera ensuite réalisée, puis la rédaction du rapport et la préparation à la soutenance seront enfin effectuées.



## Bibliographie

- [1] Text Encoding Initiative, "*12 Critical Apparatus*", P5 : Recommandations pour l'encodage et l'échange de textes électroniques, 19/08/2020. URL :  
<https://tei-c.org/release/doc/tei-p5-doc/fr/html/TC.html>
- [2] Raffaele Viglianti, "*Why TEI Stand-off Markup Authoring Needs Simplification*", Journal of the Text Encoding Initiative [Online], Issue 10 | December 2016 - July 2019, Online since 16 June 2019. URL :  
<http://journals.openedition.org/jtei/1838>; DOI :  
<https://doi.org/10.4000/jtei.1838>
- [3] Raffaele Viglianti et Daniel Rösenstrunk, "*coreBuilder*", 30/03/2017. URL :  
<https://github.com/raffazizzi/coreBuilder>

## Annexes

# A

## Cahier de spécifications

### 1 Description des interfaces externes du logiciel

#### 1.1 Interfaces homme-machine

Concernant l'ergonomie du système, un message d'erreur est généré en cas d'importation d'un fichier qui n'est pas au format XML.

The screenshot shows a web-based 'Add file' dialog. At the top, it says 'Add file' with a close button (x). Below this, there are two tabs: 'Upload' and 'Web Address (URL)'. The 'Upload' tab is active. Below the tabs is a large dashed rectangular area with the text 'Just drag and drop files here or click browse'. Below this area is a blue 'Browse' button and the text '1 files selected'. Below the 'Browse' button, there is an error message: 'Wrong file type - try with an XML file'. At the bottom right, there are two buttons: 'Close' and 'Open'.

**Figure A.1** – *Importation d'un mauvais format de fichier*

La navigation dans l'application se fait principalement par l'intermédiaire d'un menu situé à gauche de l'application.

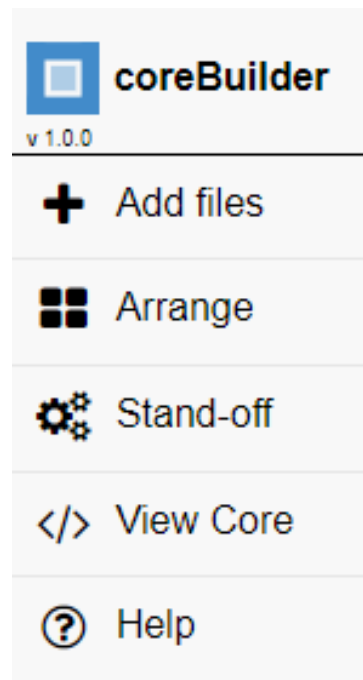


Figure A.2 – Menu de l'application

## 1.2 Interfaces logiciel-logiciel

L'application n'accède à aucun système de gestion de bases de données étant donné que les fichiers sont importés en local et que le core constituant le fichier de sortie est téléchargé également en local. Cette application ne présente pas de dépendance avec d'autres logiciels ou applications.

## 2 Spécifications fonctionnelles

Cette partie décrit l'ensemble des nouvelles fonctionnalités à intégrer à l'application actuelle.

### 2.1 Définition de la fonction de reprise d'un travail déjà en cours

#### 2.1.1 Identification de la fonction

Le rôle de la fonction de reprise d'un travail déjà en cours est de permettre à l'utilisateur de poursuivre un travail qu'il a commencé lorsqu'il relance l'application par l'importation des fichiers d'entrée ainsi que du fichier de sortie représentant le core préalablement téléchargé au format XML avant de quitter l'application.

La réalisation de cette fonction est primordiale puisqu'elle permettra aux utilisateurs de reprendre leur travail, ce qui est essentiel pour les longs travaux puisqu'actuellement l'application doit rester ouverte durant toute la durée du travail de l'utilisateur.

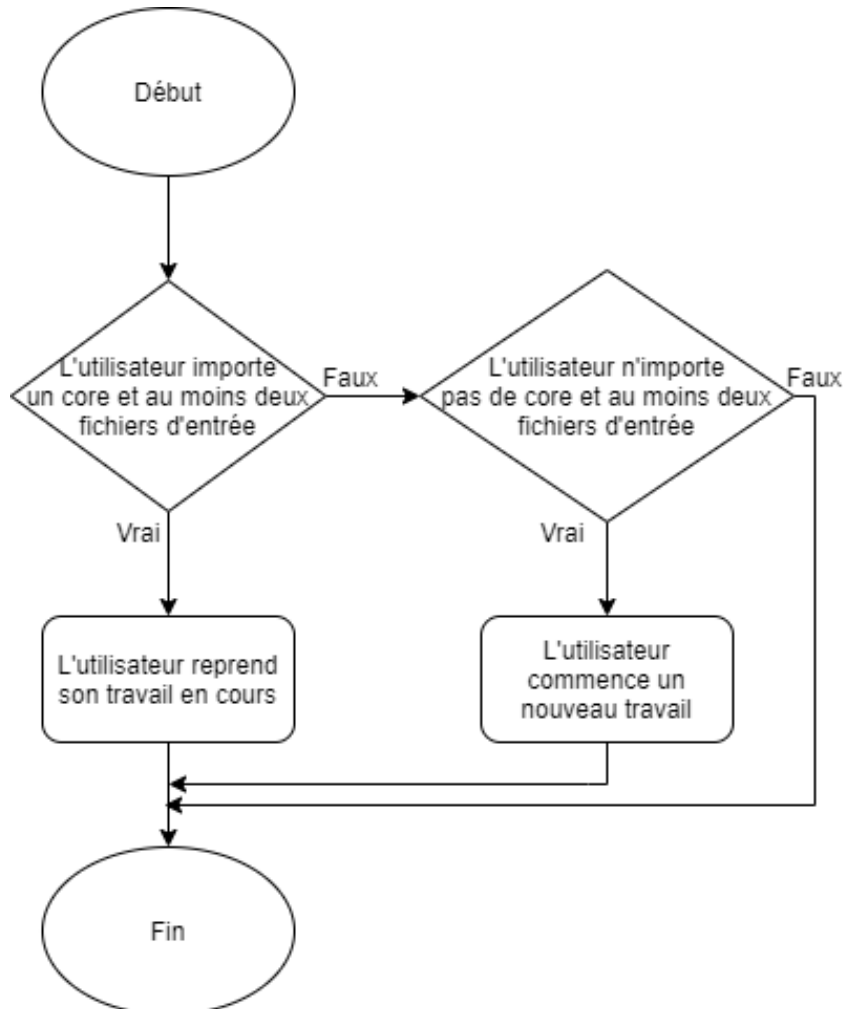
#### 2.1.2 Description de la fonction

Les entrées de la fonction sont l'ensemble des fichiers d'entrée et le fichier de sortie téléchargé au format XML au préalable et les sorties sont les fenêtres de visualisation des différents fichiers

importés. Les préconditions sont d'importer au moins deux fichiers d'entrée et le fichier de sortie téléchargé au format XML précédemment et la post-condition est d'avoir le core rempli avec le contenu du fichier de sortie importé en tant que tel.

Cette fonction interagit avec les classes "FileUploadComponent" et "ViewCoreComponent" du diagramme de la vue de l'application issu de la figure 4.6(Chapitre 4).

Le traitement associé à la fonction peut être représenté avec le diagramme suivant.



**Figure A.3** – Diagramme décrivant la fonction de reprise d'un travail déjà en cours

Celui-ci montre que dans le cas où l'utilisateur importerait un core et plusieurs fichiers d'entrée, il peut reprendre un travail en cours et dans le cas où il n'importerait pas de core, il commence un nouveau travail à partir des fichiers importés.

## 2.2 Définition de la fonction d'ajout d'un lemme

### 2.2.1 Identification de la fonction

Le rôle de la fonction d'ajout d'un lemme est de permettre à l'utilisateur d'identifier un groupe de mots de base lors de la sélection des éléments associés des différents fichiers, afin qu'il puisse par la suite définir une variation textuelle à partir de ce lemme. Cela est possible uniquement lorsque l'option concernant l'utilisation du lemme est définie par l'utilisateur au début d'un nouveau projet.



La réalisation de cette fonction est primordiale puisqu'elle permettra la réalisation de la fonction d'ajout de la variation textuelle qui est également primordiale.

### 2.2.2 Description de la fonction

L'entrée de la fonction est la balise, avec l'attribut "xml :id", sélectionnée par l'utilisateur et la sortie est une balise "lem". La précondition associée est que l'utilisateur ait choisi l'option de l'utilisation du lemme au début du projet et qu'il n'ait pas déjà choisi un lemme pour l'élément correspondant dans un des autres fichiers étant donné que chacun des groupes de mots ne peut avoir au maximum qu'un seul lemme sur lequel s'appuyer lors du choix du type de variation.

Cette fonction interagit avec les classes "SetElementsComponent" et "ViewCoreComponent" du diagramme de la vue de l'application issu de la figure 4.6 (Chapitre 4).

Le traitement associé à la fonction peut être représenté avec le diagramme suivant.

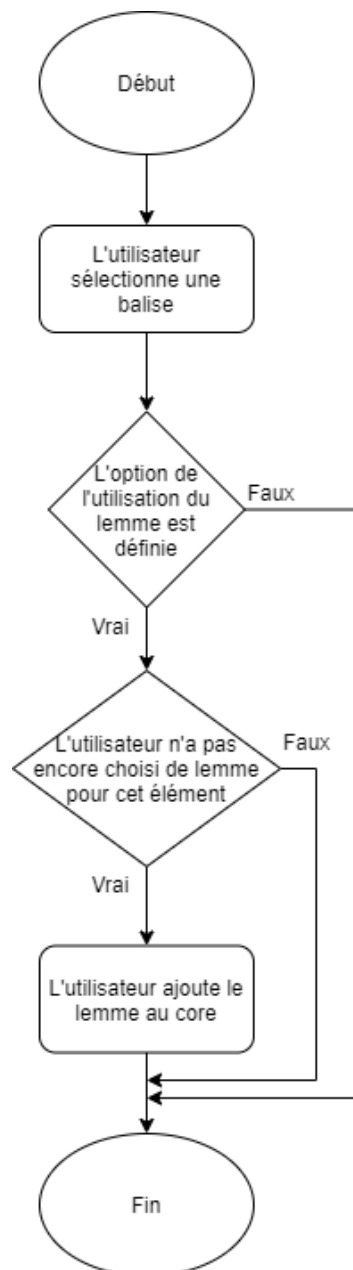


Figure A.4 – Diagramme décrivant la fonction d'ajout d'un lemme

Celui-ci montre que lorsque l'utilisateur sélectionne une balise et qu'il est dans le cadre d'un projet avec utilisation d'un lemme, si l'élément en question a déjà été défini en tant que lemme, celui-ci est ajouté en tant que lemme ou non.

Aucune gestion d'erreurs ne sera associée étant donné que l'ajout d'un élément en tant que lemme sera indisponible pour l'utilisateur s'il a déjà choisi un lemme pour celui-ci auparavant, ce qui pourrait constituer le seul cas d'erreur.

## 2.3 Définition de la fonction d'ajout du type de variation textuelle

### 2.3.1 Identification de la fonction

Le rôle de la fonction d'ajout du type de variation textuelle est de pouvoir donner la possibilité à l'utilisateur s'il le souhaite, de caractériser le type de la variation textuelle, à savoir orthographique, sémantique, de ponctuation, d'omission, de répétition ou autre qu'il peut définir, lors de la phase de groupement des différents groupes de mots sélectionnés, en fonction du lemme spécifié qui est le groupe de mots de base.

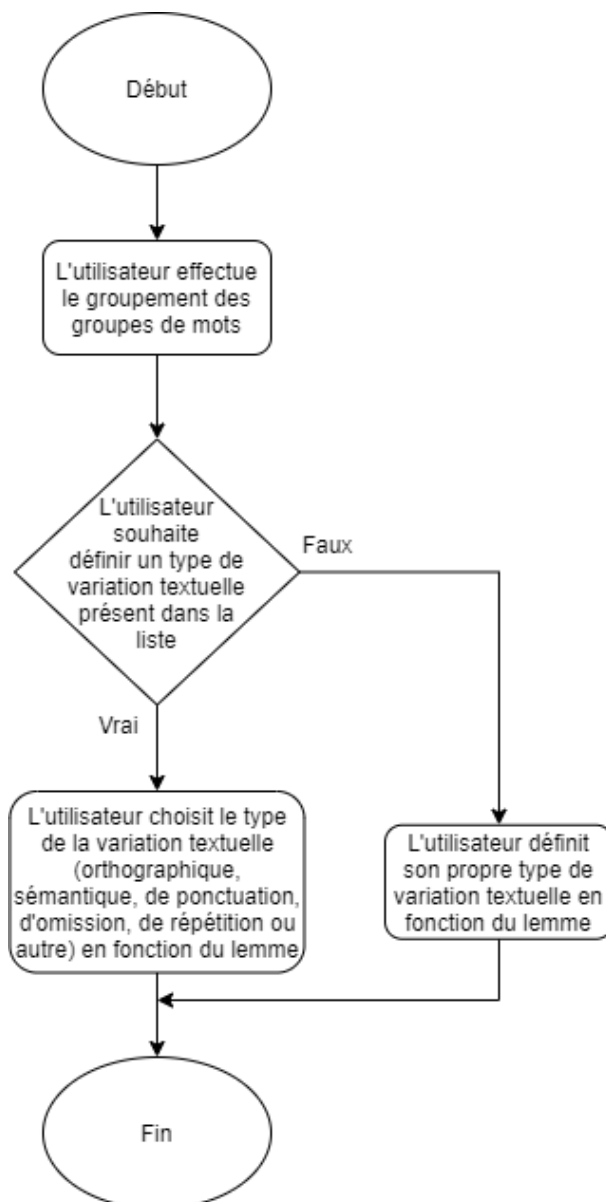
La réalisation de cette fonction est primordiale puisqu'elle permettra aux utilisateurs d'avoir éventuellement des indications supplémentaires sur le travail effectué.

### 2.3.2 Description de la fonction

Les entrées de la fonction sont les balises, avec l'attribut "xml :id", sélectionnées par l'utilisateur et la sortie est une balise comprenant le groupement avec le type de variation textuelle précisé. Les préconditions associées sont que l'utilisateur ait préalablement sélectionné plusieurs balises dans les différents fichiers XML et qu'il ait établi un lemme à partir duquel effectuer le groupement. La réalisation de cette fonction dépend donc de la réalisation de la précédente d'ajout d'un lemme. La post-condition est d'avoir un type de variation textuelle pour le groupement effectué, qui est orthographique, sémantique, de ponctuation, d'omission, de répétition ou autre. Cette fonction dépend directement de la fonction d'ajout du lemme.

Cette fonction interagit avec les classes "SetElementsComponent" et "ViewCoreComponent" du diagramme de la vue de l'application issu de la figure 4.6 (Chapitre 4).

Le traitement associé à la fonction peut être représenté avec le diagramme suivant.



**Figure A.5** – Diagramme décrivant la fonction d'ajout du type de variation textuelle

Celui-ci montre que l'utilisateur est tout d'abord amené à effectuer un groupement des groupes de mots qui présentent les mêmes types de variation textuelle par rapport au lemme, puis à choisir ce type s'il le souhaite.

## 2.4 Définition de la fonction d'ajout d'une fenêtre de visualisation du core en cours

### 2.4.1 Identification de la fonction

Le rôle de la fonction d'ajout d'une fenêtre de visualisation du core en cours est de permettre à l'utilisateur de visualiser en temps réel l'évolution de son travail, c'est-à-dire de voir l'ensemble du balisage réalisé sans avoir à passer par le menu et une autre fenêtre. Une visualisation au format HTML par l'utilisation d'un fichier CSS sera ajoutée à la visualisation au format XML actuelle. Cela entraînera la suppression de l'onglet "View Core" dans le menu de l'application existante puisque celui-ci n'aura plus d'utilité. En effet, le core sera visible directement sur la

même page que les fichiers d'entrée et il sera également possible de télécharger le core depuis celle-ci.

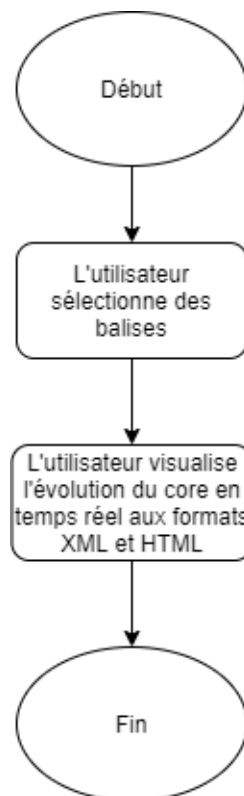
La réalisation de cette fonction est primordiale puisqu'elle conférera aux utilisateurs un gain de praticité au cours de l'évolution de leur travail.

### 2.4.2 Description de la fonction

Les entrées de la fonction sont l'ensemble des éléments sélectionnés par l'utilisateur et la sortie est une fenêtre contenant le balisage associé avec les formats XML et HTML présents.

Cette fonction interagit avec les classes "SetElementsComponent" et "ViewCoreComponent" du diagramme de la vue de l'application issu de la figure 4.6 (Chapitre 4).

Le traitement associé à la fonction peut être représenté avec le diagramme suivant.



**Figure A.6** – *Diagramme décrivant la fonction d'ajout d'une fenêtre de visualisation du core en cours*

Celui-ci montre que l'utilisateur peut visualiser de façon presque instantanée les répercussions de la sélection des balises sur le core.

## 2.5 Définition de la fonction d'ajout de formats de téléchargement du core

### 2.5.1 Identification de la fonction

Le rôle de la fonction d'ajout de formats de téléchargement du core est de permettre à l'utilisateur de télécharger le core aux formats HTML et PDF en plus de XML. La visualisation du core au format XML sur l'application actuelle est conservée.

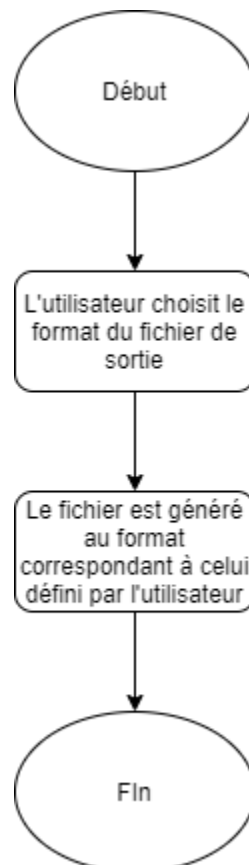
La réalisation de cette fonction est primordiale puisqu'elle permettra aux utilisateurs de disposer de leur travail réalisé dans différents formats suivant leurs besoins par la suite.

### 2.5.2 Description de la fonction

L'entrée de la fonction est le core réalisé par l'utilisateur au format XML et la sortie est le fichier de sortie au format HTML ou PDF tel que défini par l'utilisateur. La post-condition est que le fichier de sortie soit au format HTML ou PDF, mais il n'y a pas de précondition particulière dans ce cas de figure.

Cette fonction interagit avec la classe "ViewCoreComponent" du diagramme de la vue de l'application issu de la figure 4.6 (Chapitre 4).

Le traitement associé à la fonction peut être représenté avec le diagramme suivant.



**Figure A.7** – Diagramme décrivant la fonction d'ajout de formats de téléchargement du core

Celui-ci montre que l'utilisateur choisit le format de fichier qu'il désire avant que le fichier ne lui soit généré au format correspondant.

## 2.6 Définition de la fonction du changement des couleurs des types de variation textuelle

### 2.6.1 Identification de la fonction

Le rôle de la fonction de changement des couleurs des types de variation textuelle par le biais d'une palette de couleurs est de permettre à l'utilisateur de personnaliser les couleurs affectées aux différents types de variation textuelle, à savoir orthographique, sémantique, de ponctuation, d'omission, de répétition ou autre, en sélectionnant les couleurs souhaitées pour chacun des types de variation textuelle depuis une palette de couleurs.

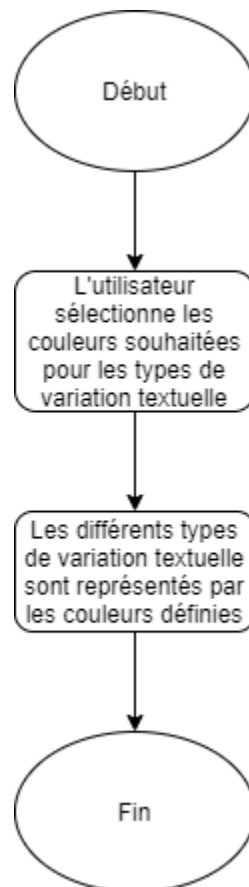
La réalisation de cette fonction est secondaire puisque les fonctions définies précédemment sont absolument nécessaires pour pouvoir utiliser correctement l'application contrairement à celle-ci.

### 2.6.2 Description de la fonction

Les entrées de la fonction sont les couleurs définies actuellement dans l'application pour les différents types de variation textuelle en sachant que par défaut, les variations orthographiques seront en vert, les variations sémantiques en rouge, les variations de ponctuation en orange, les variations d'omission en bleu et les variations de répétition en violet. Les post-conditions sont que les différents types de variation textuelle soient représentés par les couleurs sélectionnées par l'utilisateur.

Cette fonction interagit avec la classe "SetElementsComponent" du diagramme de la vue de l'application issu de la figure 4.6 (Chapitre 4).

Le traitement associé à la fonction peut être représenté avec le diagramme suivant.



**Figure A.8** – *Diagramme décrivant la fonction du changement des couleurs des types de variation textuelle*

Celui-ci montre que l'utilisateur sélectionne tout d'abord les couleurs qu'il souhaite pour les différents types de variation textuelle, puis les variations textuelles sont identifiées par les couleurs adéquates.

## 2.7 Définition de la fonction d'ajout de différents types de visualisation des fichiers d'entrée

### 2.7.1 Identification de la fonction

Le rôle de la fonction d'ajout de différents types de visualisation des fichiers d'entrée est de permettre à l'utilisateur de choisir le type de visualisation souhaité dans les fenêtres contenant ces fichiers entre la visualisation par le code qui est la visualisation par défaut et la visualisation par le texte.

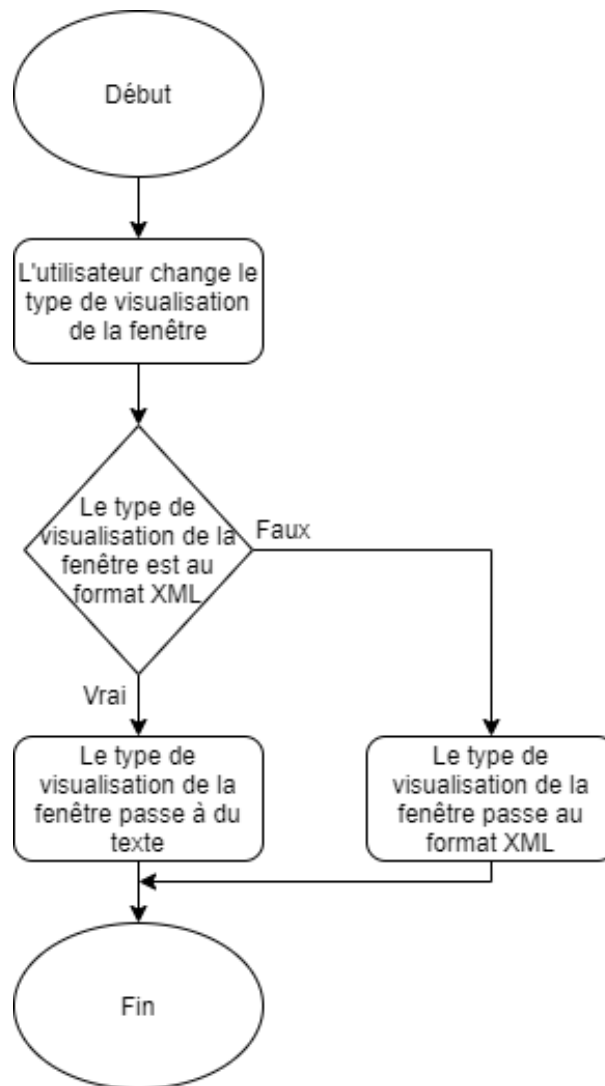
La réalisation de cette fonction est secondaire pour la même raison que précédemment.

### 2.7.2 Description de la fonction

Les entrées de la fonction sont les contenus des fichiers XML et les sorties sont des contenus au format XML ou au format HTML en prenant en compte les attributs CSS présents dans les balises "rendition" des fichiers XML, en fonction de la sélection de l'utilisateur. La post-condition est que le contenu de la fenêtre soit au format XML ou texte en fonction du choix de l'utilisateur.

Cette fonction interagit avec la classe "FileUploadComponent" du diagramme de la vue de l'application issu de la figure 4.6(Chapitre 4).

Le traitement associé à la fonction peut être représenté avec le diagramme suivant.



**Figure A.9** – Diagramme décrivant la fonction d'ajout de différents types de visualisation des fichiers d'entrée

Celui-ci montre que lorsque l'utilisateur change le type de visualisation de la fenêtre, si celui-ci est au format XML, il passe à du texte, sinon au format XML.

## 2.8 Définition de la fonction de répercussion de l'état du scrolling

### 2.8.1 Identification de la fonction

Le rôle de la fonction de répercussion de l'état du scrolling est de permettre à l'utilisateur lorsqu'il scrolle dans une fenêtre d'avoir les mêmes états de scrolling dans les autres fenêtres que dans celle-ci.

La réalisation de cette fonction est facultative puisqu'elle n'a pas été exprimée par la cliente.

### 2.8.2 Description de la fonction

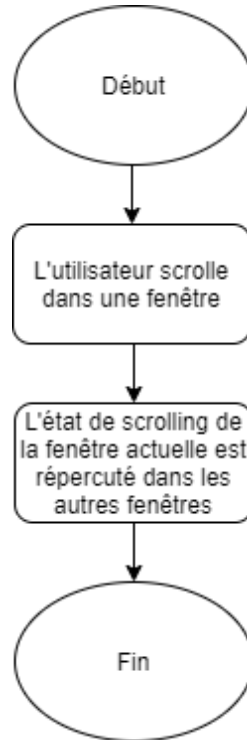
L'entrée de la fonction est l'état du scrolling de la fenêtre actuelle et les sorties sont des états de scrolling identiques à celui de la fenêtre actuelle. La précondition est que l'utilisateur scrolle dans



une fenêtre et les post-conditions sont que les états de scrolling des autres fenêtres correspondent à celui de la fenêtre actuelle.

Cette fonction interagit avec la classe "SetElementsComponent" du diagramme de la vue de l'application issu de la figure 4.6 (Chapitre 4).

Le traitement associé à la fonction peut être représenté avec le diagramme suivant.



**Figure A.10** – *Diagramme décrivant la fonction de répercussion de l'état du scrolling*

Celui-ci montre que lorsque l'utilisateur scrolle dans une des fenêtres, cet état de scrolling est répercuté dans les autres fenêtres.

### 3 Spécifications non fonctionnelles

#### 3.1 Contraintes de développement et conception

Pour le développement de l'application, l'éditeur de code Visual Studio Code est utilisé sous le système d'exploitation Windows.

Les langages utilisés pour le développement sont HTML pour structurer les pages Web, Sass pour générer les feuilles de style nécessaires à la mise en forme des pages Web, JavaScript pour l'interactivité des pages Web et XSLT pour transformer les fichiers XML en fichiers HTML et PDF.

Les plateformes logicielles utilisées sont Node.js qui est un environnement d'exécution JavaScript permettant la compilation et l'exécution du code JavaScript et gulp.js permettant l'automatisation de tâches.

### 3.2 Contraintes de fonctionnement et d'exploitation

#### 3.2.1 Performances

Au niveau des spécifications en temps réel qui sont liées à l'utilisation du système, d'un point de vue de l'utilisateur, celui-ci sera amené à être utilisé fréquemment et d'un point de vue de l'environnement, il n'y a aucune dépendance entre le système et l'environnement puisque c'est une application Web accessible depuis n'importe quel navigateur Web.

#### 3.2.2 Sécurité

Le système n'est utilisé que par un seul type d'utilisateurs. De plus, l'application Web est open source et par conséquent aucun contrôle d'accès n'est mis en place au niveau de l'application, elle est complètement libre de droits.

# B

# Cahier du développeur

## 1 Introduction

Cette partie permet de décrire l'ensemble des aspects techniques du projet.

## 2 Conventions de développement

Le code développé doit respecter certaines conventions.

Chaque classe doit se situer dans un seul et unique fichier. Les noms de classes doivent être en UpperCamelCase, c'est-à-dire que chacun des mots doit être écrit à la suite sans séparateur en commençant par une lettre majuscule pour chacun d'entre eux et en anglais. Les noms de constantes définies dans ces classes doivent être en majuscules avec comme séparateur le symbole underscore et en anglais.

Les noms de méthodes, fonctions et variables doivent être en camelCase, c'est-à-dire que chacun des mots doit être écrit à la suite sans séparateur en commençant par une lettre majuscule pour chacun d'entre eux sauf pour le premier et en anglais.

## 3 Diagramme de classes du système

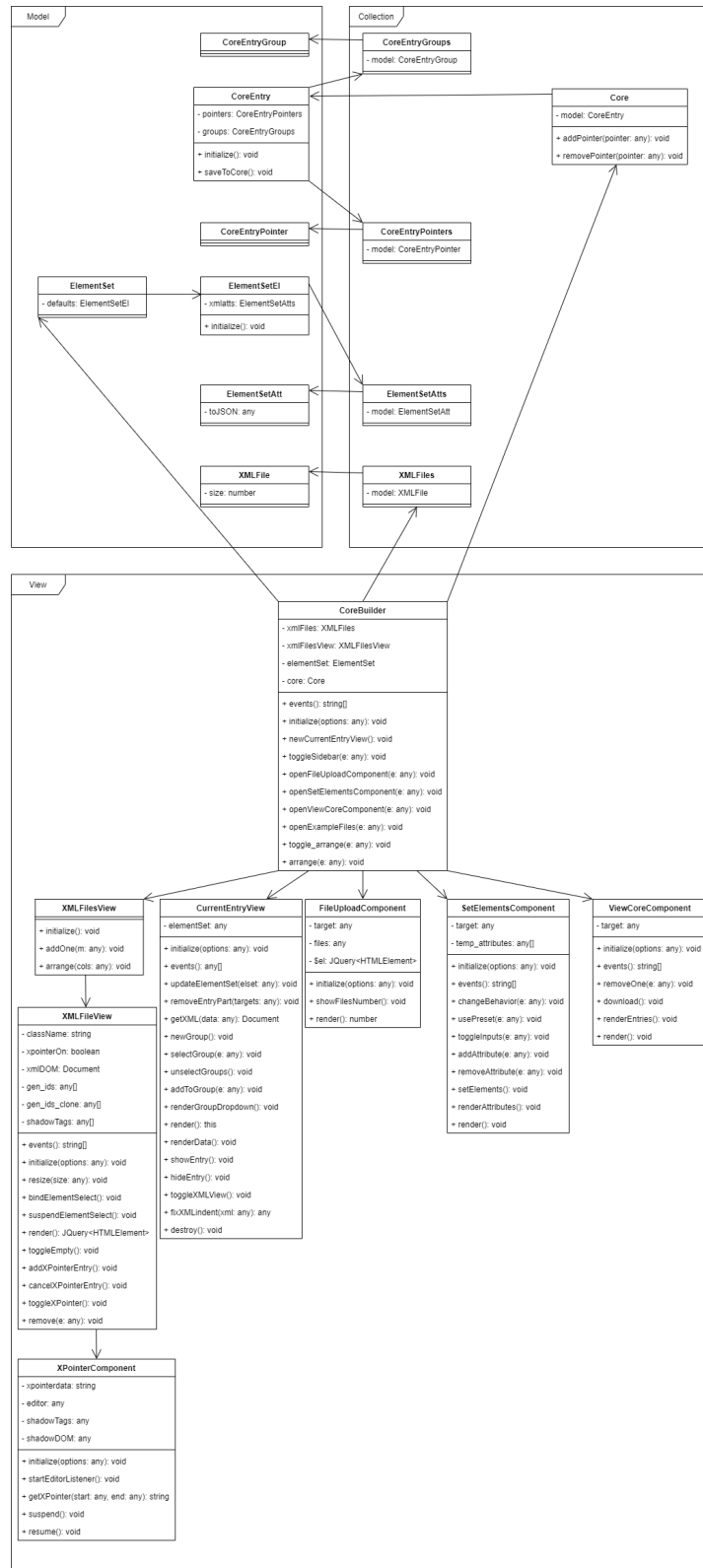


Figure B.1 – Diagramme de classes du système

Ce diagramme de classes global et détaillé représente l'architecture de l'application. Celle-ci se compose de trois parties, une pour les modèles, une pour les vues et une autre pour les collections.

## 4 Description détaillée des classes

**La partie des modèles comporte plusieurs classes.**

- La classe "CoreEntryGroup" représente un groupement effectué dans le core.
- La classe "CoreEntry" représente une entrée d'apparat critique du core.
  - Elle comporte plusieurs attributs.
    - L'attribut "pointers" de type "CoreEntryPointers" représente l'ensemble des expressions créées dans le core par la sélection de texte directement dans les fichiers XML.
    - L'attribut "groups" de type "CoreEntryGroups" représente l'ensemble des groupements effectués dans le core.
  - Elle comporte également plusieurs méthodes.
    - La méthode "initialize()" initialise les attributs par l'instanciation des objets de type "CoreEntryPointers" et "CoreEntryGroups" à partir de leurs constructeurs.
    - La méthode "saveToCore()" ajoute une nouvelle entrée d'apparat critique au core.
- La classe "CoreEntryPointer" représente une expression créée dans le core par la sélection de texte directement dans un fichier XML.
- La classe "ElementSet" représente les éléments de balisage stand-off par défaut.
  - Elle comporte un attribut.
    - L'attribut "defaults" de type "ElementSetEl" représente les attributs XML en fonction du type de balisage stand-off par défaut.
- La classe "ElementSetEl" représente les attributs XML en fonction du type de balisage stand-off sélectionné par l'utilisateur.
  - Elle comporte un attribut.
    - L'attribut "xmlatts" de type "ElementSetAtts" représente l'ensemble des attributs XML sélectionnés par l'utilisateur dans les fichiers XML.
  - Elle comporte également une méthode.
    - La méthode "initialize()" initialise l'attribut par l'instanciation de l'objet de type "ElementSetAtts" à partir de son constructeur.
- La classe "ElementSetAtt" représente l'attribut XML sélectionné par l'utilisateur dans un fichier XML.
  - Elle comporte un attribut.
    - L'attribut "toJSON" de type "any" représente l'attribut XML sélectionné par l'utilisateur dans un fichier XML étendu au format JSON.
- La classe "XMLFile" représente un fichier XML.
  - Elle comporte un attribut.
    - L'attribut "size" de type "number" représente la taille par défaut d'un fichier XML.

**La partie des vues comporte plusieurs classes.**

- La classe "CoreBuilder" gère les interactions principales avec l'utilisateur de l'application.
  - Elle comporte plusieurs attributs.
    - L'attribut "xmlFiles" de type "XMLFiles" représente l'ensemble des fichiers XML.
    - L'attribut "xmlFilesView" de type "XMLFilesView" représente les interactions de l'ensemble des fichiers XML.
    - L'attribut "elementSet" de type "ElementSet" représente les éléments de balisage stand-off par défaut.
    - L'attribut "core" de type "Core" représente l'ensemble des entrées d'apparat critique du core.
  - Elle comporte également plusieurs méthodes.
    - La méthode "events()" renvoie le hachage des événements qui associe les événements aux méthodes de la vue sous la forme d'un tableau de "string".
    - La méthode "initialize(options : any)" prend en paramètre des "options" de type "any" et initialise les attributs par l'instanciation des objets de type "XMLFiles",

- "XMLFilesView", "ElementSet" et "Core" à partir de leurs constructeurs.
- La méthode "newCurrentEntryView()" instancie un objet de type "CurrentEntryView" permettant de gérer les interactions avec la fenêtre de l'entrée d'apparat critique en cours.
  - La méthode "toggleSidebar(e : any)" prend en paramètre un événement "e" de type "any" et permute l'affichage de la barre latérale contenant le menu.
  - La méthode "openFileUploadComponent(e : any)" prend en paramètre un événement "e" de type "any" et instancie un objet de type "FileUploadComponent" permettant de gérer les interactions lors de l'importation des fichiers.
  - La méthode "openSetElementsComponent(e : any)" prend en paramètre un événement "e" de type "any" et instancie un objet de type "SetElementsComponent" permettant de gérer les interactions lors de la définition des éléments de balisage stand-off.
  - La méthode "openViewCoreComponent(e : any)" prend en paramètre un événement "e" de type "any" et instancie un objet de type "ViewCoreComponent" permettant de gérer les interactions du pop-up de visualisation du core.
  - La méthode "openExampleFiles(e : any)" prend en paramètre un événement "e" de type "any" et ouvre les fichiers XML d'entrée d'exemple.
  - La méthode "toggle\_arrange(e : any)" prend en paramètre un événement "e" de type "any" et détecte la disposition des fenêtres contenant les fichiers XML d'entrée choisie par l'utilisateur.
  - La méthode "arrange(e : any)" prend en paramètre un événement "e" de type "any" et permute la disposition des fenêtres contenant les fichiers XML d'entrée en fonction du choix fait par l'utilisateur.
  - La classe "XMLFilesView" gère les interactions de l'ensemble des fichiers XML.
    - Elle comporte plusieurs méthodes.
      - La méthode "initialize()" écoute les événements qui correspondent à l'ajout ou à la suppression d'un fichier XML d'entrée.
      - La méthode "addOne(m : any)" prend en paramètre la disposition "m" du fichier XML d'entrée et l'ajoute à ceux présents actuellement.
      - La méthode "arrange(cols : any)" prend en paramètre les colonnes "cols" de la disposition des fichiers XML d'entrée et permute ces fichiers suite à l'ajout ou à la suppression de l'un d'entre eux.
  - La classe "XMLFileView" gère les interactions spécifiques à chacun des fichiers XML.
    - Elle comporte plusieurs attributs.
      - L'attribut "className" de type "string" représente le nom de classe HTML du fichier XML.
      - L'attribut "xpointerOn" de type "boolean" représente l'activation ou non du mode permettant de créer des expressions dans le core par la sélection de texte directement dans un fichier XML.
      - L'attribut "xmlDOM" de type "Document" représente l'analyse du code XML.
      - L'attribut "gen\_ids" de type tableau de "any" représente les nouveaux identifiants XML.
      - L'attribut "gen\_ids\_clone" de type tableau de "any" représente une copie de l'attribut "gen\_ids".
      - L'attribut "shadowTags" de type tableau de "any" représente les balises implicites sélectionnées par l'utilisateur lors de la création d'expressions dans le core.
    - Elle comporte également plusieurs méthodes.
      - La méthode "events()" renvoie le hachage des événements qui associe les événements aux méthodes de la vue sous la forme d'un tableau de "string".
      - La méthode "initialize(options : any)" prend en paramètre des "options" de type "any" et initialise les attributs.

- La méthode "resize(size : any)" prend en paramètre une taille "size" de type "any" et supprime et ajoute des classes HTML.
- La méthode "bindElementSelect()" effectue la liaison avec l'élément sélectionné par l'utilisateur.
- La méthode "suspendElementSelect()" suspend la sélection des éléments par l'utilisateur.
- La méthode "render()" renvoie le fichier avec les balises mises à jour sous la forme "JQery<HTMLElement>".
- La méthode "toggleEmpty()" permute le mode permettant de sélectionner des éléments dans les fichiers XML d'entrée et celui ne le permettant pas.
- La méthode "addXPointerEntry()" ajoute une expression créée dans un fichier XML d'entrée au core.
- La méthode "cancelXPointerEntry()" permet de passer au mode ne permettant pas de créer des expressions dans les fichiers XML d'entrée.
- La méthode "toggleXPointer()" permet de permuter entre le mode permettant de créer des expressions dans les fichiers XML d'entrée et celui ne le permettant pas.
- La méthode "remove(e : any)" prend en paramètre un événement "e" de type "any" permettant de retirer un fichier XML d'entrée.
- La classe "XPointerComponent" gère les interactions concernant la création d'expressions dans les fichiers XML d'entrée par l'utilisateur.
  - Elle comporte plusieurs attributs.
    - L'attribut "xpointerdata" de type "string" représente l'expression créée dans un fichier XML d'entrée par l'utilisateur.
    - L'attribut "editor" de type "any" représente la sélection de l'utilisateur.
    - L'attribut "shadowTags" de type "any" représente les balises sélectionnées par l'utilisateur.
    - L'attribut "shadowDOM" de type "any" représente le document XML.
  - Elle comporte également plusieurs méthodes.
    - La méthode "initialize(options : any)" prend en paramètre des "options" de type "any" et initialise les attributs.
    - La méthode "startEditorListener()" écoute les interactions faites par l'utilisateur.
    - La méthode "getXPointer(start : any, end : any)" prend en paramètre le début "start" de type "any" et la fin "end" de type "any" de l'expression créée par l'utilisateur dans un fichier XML d'entrée et la renvoie sous forme de "string".
    - La méthode "suspend()" arrête la sélection de l'expression lorsque l'utilisateur relâche le bouton de la souris.
    - La méthode "resume()" permet à l'utilisateur de valider ou non la sélection de l'expression effectuée.
- La classe "CurrentEntryView" gère les interactions avec la fenêtre de l'entrée d'apparat critique en cours.
  - Elle comporte un attribut.
    - L'attribut "elementSet" de type "any" représente l'élément sélectionné par l'utilisateur.
  - Elle comporte également plusieurs méthodes.
    - La méthode "initialize(options : any)" prend en paramètre des "options" de type "any" et initialise l'attribut.
    - La méthode "events()" renvoie le hachage des événements qui associe les événements aux méthodes de la vue sous la forme d'un tableau de "any".
    - La méthode "updateElementSet(elset : any)" prend en paramètre l'élément "elset" de type "any" sélectionné par l'utilisateur et vérifie qu'il n'a pas déjà été sélectionné.
    - La méthode "removeEntryPart(targets : any)" prend en paramètre les cibles "targets" de type "any" et supprime une entrée d'apparat critique en fonction du

- paramètre.
- La méthode "getXML(data : any)" prend en paramètre l'entrée d'apparat critique "data" de type "any" et renvoie l'entrée d'apparat critique sous forme de "Document" pour récupérer le format XML.
- La méthode "newGroup()" crée un nouveau groupement pour le core.
- La méthode "selectGroup(e : any)" prend en paramètre un événement "e" de type "any" et sélectionne un groupement.
- La méthode "unselectGroups()" désélectionne l'ensemble des groupements.
- La méthode "addToGroup(e : any)" prend en paramètre un événement "e" de type "any" et ajoute au groupement les éléments sélectionnés par l'utilisateur.
- La méthode "renderGroupDropdown()" permute entre une vue détaillée et une vue non détaillée du groupement en fonction du choix de l'utilisateur.
- La méthode "render()" renvoie la référence "this" auquel l'objet appartient.
- La méthode "renderData()" met en forme les données contenues dans le core.
- La méthode "showEntry()" rend visible l'entrée d'apparat critique.
- La méthode "hideEntry()" permet de cacher l'entrée d'apparat critique.
- La méthode "toggleXMLView()" permet de permuter entre l'affichage au format XML et l'affichage standard.
- La méthode "fixXMLindent(xml : any)" prend en paramètre un code "xml" de type "any" et le renvoie indenté sous forme de "any".
- La méthode "destroy()" supprime une entrée d'apparat critique.
- La classe "FileUploadComponent" gère les interactions lors de l'importation des fichiers.
  - Elle comporte plusieurs attributs.
    - L'attribut "target" de type "any" représente la cible du fichier importé.
    - L'attribut "files" de type "any" représente les fichiers.
    - L'attribut "\$el" de type "jQuery<HTMLElement>" représente l'élément JQuery du fichier XML.
  - Elle comporte également plusieurs méthodes.
    - La méthode "initialize(options : any)" prend en paramètre des "options" de type "any" et initialise les attributs.
    - La méthode "showFilesNumber()" affiche le nombre de fichiers importés.
    - La méthode "render()" retourne un "number" indiquant si le fichier importé est supporté ou non par le navigateur Web.
- La classe "SetElementsComponent" gère les interactions lors de la définition des éléments de balisage stand-off.
  - Elle comporte plusieurs attributs.
    - L'attribut "target" de type "any" représente la cible des éléments de balisage stand-off sélectionnés.
    - L'attribut "temp\_attributes" de type tableau de "any" représente l'ensemble des attributs temporaires.
  - Elle comporte également plusieurs méthodes.
    - La méthode "initialize(options : any)" prend en paramètre des "options" de type "any" et initialise les attributs.
    - La méthode "events()" renvoie le hachage des événements qui associe les événements aux méthodes de la vue sous la forme d'un tableau de "string".
    - La méthode "changeBehavior(e : any)" prend en paramètre un événement "e" de type "any" et change le comportement des éléments de balisage stand-off.
    - La méthode "usePreset(e : any)" prend en paramètre un événement "e" de type "any" et change les éléments de balisage stand-off en fonction du choix de l'utilisateur.
    - La méthode "toggleInputs(e : any)" prend en paramètre un événement "e" de type "any" et permute les entrées des éléments.
    - La méthode "addAttribute(e : any)" prend en paramètre un événement "e" de



- type "any" et ajoute un attribut.
- La méthode "removeAttribute(e : any)" prend en paramètre un événement "e" de type "any" et supprime un attribut.
- La méthode "setElements()" sélectionne les éléments de balisage stand-off.
- La méthode "renderAttributes()" transforme les attributs sélectionnés en attributs de type XML. La méthode "render()" met en forme l'ensemble des éléments de balisage stand-off.
- La classe "ViewCoreComponent" gère les interactions du pop-up de visualisation du core.
  - Elle comporte un attribut.
    - L'attribut "target" de type "any" représente la cible de la fenêtre de visualisation du core.
  - Elle comporte également plusieurs méthodes.
    - La méthode "initialize(options : any)" prend en paramètre des "options" de type "any" et initialise l'attribut.
    - La méthode "events()" renvoie le hachage des événements qui associe les événements aux méthodes de la vue sous la forme d'un tableau de "string".
    - La méthode "removeOne(e : any)" prend en paramètre un événement "e" de type "any" et supprime une entrée d'apparat critique du core.
    - La méthode "download()" télécharge le core au format XML.
    - La méthode "renderEntries()" affiche les entrées d'apparat critique du core dans la fenêtre de visualisation.
    - La méthode "render()" affiche l'ensemble des entrées d'apparat critique du core dans la fenêtre de visualisation.

#### La partie des collections comporte plusieurs classes.

- La classe "CoreEntryGroups" regroupe les modèles "CoreEntryGroup" et représente donc l'ensemble des groupements effectués dans le core.
  - Elle comporte un attribut.
    - L'attribut "model" de type "CoreEntryGroup" représente un groupement effectué dans le core.
- La classe "Core" regroupe les modèles "CoreEntry" et représente donc l'ensemble des entrées d'apparat critique du core.
  - Elle comporte un attribut.
    - L'attribut "model" de type "CoreEntry" représente une entrée d'apparat critique du core.
  - Elle comporte également plusieurs méthodes.
    - La méthode "addPointer(pointer : any)" prend en paramètre la dernière expression "pointer" de type "any" créée par la sélection de texte dans un fichier XML par l'utilisateur, prend la dernière entrée d'apparat critique et l'ajoute.
    - La méthode "removePointer(pointer : any)" prend en paramètre la dernière expression "pointer" de type "any" créée par la sélection de texte dans un fichier XML par l'utilisateur, prend la dernière entrée d'apparat critique et cherche un pointeur à supprimer.
- La classe "CoreEntryPointers" regroupe les modèles "CoreEntryPointer" et représente donc l'ensemble des expressions créées dans le core par la sélection de texte directement dans les fichiers XML.
  - Elle comporte un attribut.
    - L'attribut "model" de type "CoreEntryPointer" représente une expression créée dans le core par la sélection de texte directement dans un fichier XML.
- La classe "ElementSetAtts" regroupe les modèles "ElementSetAtt" et représente donc l'ensemble des attributs XML sélectionnés par l'utilisateur dans les fichiers XML.

- Elle comporte un attribut.
  - L'attribut "model" de type "ElementSetAtt" représente l'attribut XML sélectionné par l'utilisateur dans un fichier XML.
- La classe "XMLFiles" regroupe les modèles "XMLFile" et représente donc l'ensemble des fichiers XML.
  - Elle comporte un attribut.
    - L'attribut "model" de type "XMLFile" représente un fichier XML.

## 1 Planification

Afin de réaliser la planification du projet pour la partie de la recherche, le diagramme de Gantt suivant est établi.

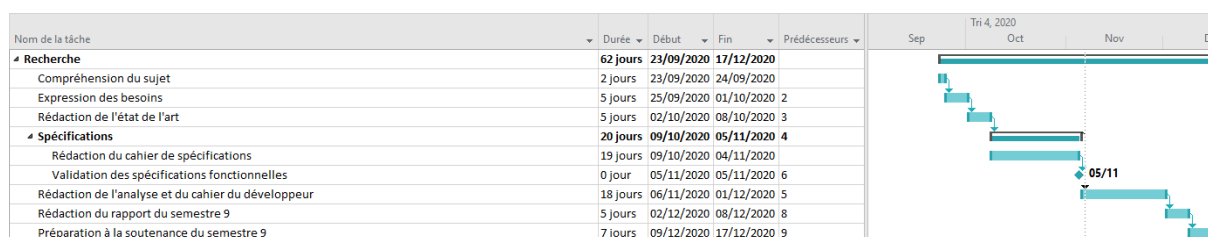


Figure C.1 – Planning du semestre 9

De la même manière, le diagramme de Gantt concernant le développement du projet est réalisé.

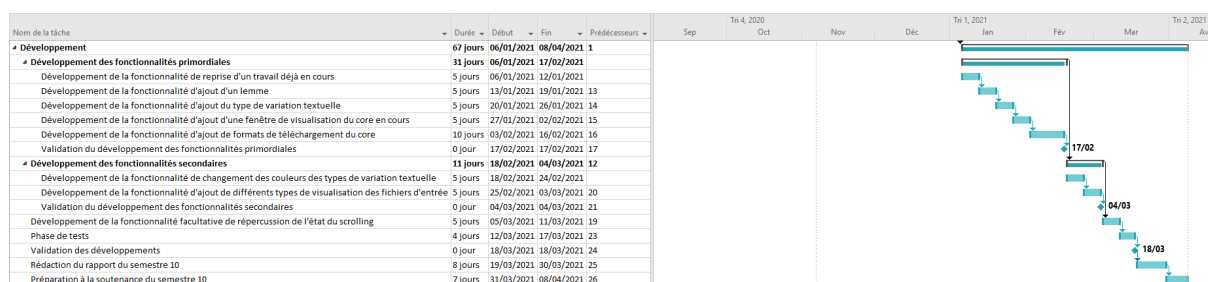


Figure C.2 – Planning du semestre 10

## 2 Méthodologie

Conformément au diagramme de Gantt précédent, le modèle du cycle en V est utilisé avec un flux descendant comportant les différentes activités de recherche jusqu'au développement des fonctionnalités et un flux ascendant intégrant les différentes tâches de finalisation du projet.

### 3 Description des tâches

#### Tâche 1 : compréhension du sujet

- Date de début : 23/09/2020
- Date de fin : 24/09/2020
- Durée : 2 jours
- Description : cette tâche consiste à comprendre le contexte dans lequel le projet s'inscrit.

#### Tâche 2 : expression des besoins

- Date de début : 25/09/2020
- Date de fin : 01/10/2020
- Durée : 5 jours
- Description : cette tâche consiste à définir les nouvelles fonctionnalités du projet à partir des besoins exprimés par la cliente.

#### Tâche 3 : rédaction de l'état de l'art

- Date de début : 02/10/2020
- Date de fin : 08/10/2020
- Durée : 5 jours
- Description : cette tâche consiste à faire état des différentes techniques de balisage de texte existantes.

#### Tâche 4 : rédaction du cahier de spécifications

- Date de début : 09/10/2020
- Date de fin : 04/11/2020
- Durée : 19 jours
- Description : cette tâche consiste à définir les spécifications du projet en lien avec les besoins exprimés par la cliente.
- Livrables : spécifications fonctionnelles

#### Tâche 5 : rédaction de l'analyse et du cahier du développeur

- Date de début : 06/11/2020
- Date de fin : 01/12/2020
- Durée : 18 jours
- Description : cette tâche consiste à définir les éléments relatifs à la modélisation de l'application.

#### Tâche 6 : rédaction du rapport du semestre 9

- Date de début : 02/12/2020
- Date de fin : 08/12/2020
- Durée : 5 jours
- Description : cette tâche consiste à finir de rédiger le rapport du semestre 9.

#### Tâche 7 : préparation à la soutenance du semestre 9

- Date de début : 09/12/2020
- Date de fin : 17/12/2020
- Durée : 7 jours
- Description : cette tâche consiste à réaliser le support de présentation et à préparer la soutenance du semestre 9.

**Tâche 8 : développement de la fonctionnalité de reprise d'un travail déjà en cours**

- Date de début : 06/01/2021
- Date de fin : 12/01/2021
- Durée : 5 jours
- Description : cette tâche consiste à développer la fonctionnalité de reprise d'un travail déjà en cours.

**Tâche 9 : développement de la fonctionnalité d'ajout d'un lemme**

- Date de début : 13/01/2021
- Date de fin : 19/01/2021
- Durée : 5 jours
- Description : cette tâche consiste à développer la fonctionnalité d'ajout d'un lemme.

**Tâche 10 : développement de la fonctionnalité d'ajout du type de variation textuelle**

- Date de début : 20/01/2021
- Date de fin : 26/01/2021
- Durée : 5 jours
- Description : cette tâche consiste à développer la fonctionnalité d'ajout du type de variation textuelle.

**Tâche 11 : développement de la fonctionnalité d'ajout d'une fenêtre de visualisation du core en cours**

- Date de début : 27/01/2021
- Date de fin : 02/02/2021
- Durée : 5 jours
- Description : cette tâche consiste à développer la fonctionnalité d'ajout d'une fenêtre de visualisation du core en cours.

**Tâche 12 : développement de la fonctionnalité d'ajout de formats de téléchargement du core**

- Date de début : 03/02/2021
- Date de fin : 16/02/2021
- Durée : 10 jours
- Description : cette tâche consiste à développer la fonctionnalité d'ajout de formats de téléchargement du core.
- Livrable : intégration des fonctionnalités primordiales à l'application

**Tâche 13 : développement de la fonctionnalité de changement des couleurs des types de variation textuelle**

- Date de début : 18/02/2021
- Date de fin : 24/02/2021
- Durée : 5 jours
- Description : cette tâche consiste à développer la fonctionnalité de changement des couleurs des types de variation textuelle.

**Tâche 14 : développement de la fonctionnalité d'ajout de différents types de visualisation des fichiers d'entrée**

- Date de début : 25/02/2021
- Date de fin : 03/03/2021
- Durée : 5 jours

- Description : cette tâche consiste à développer la fonctionnalité d'ajout de différents types de visualisation des fichiers d'entrée.
- Livrable : intégration des fonctionnalités secondaires à l'application

#### **Tâche 15 : développement de la fonctionnalité facultative de répercussion de l'état du scrolling**

- Date de début : 05/03/2021
- Date de fin : 11/03/2021
- Durée : 5 jours
- Description : cette tâche consiste à développer la fonctionnalité facultative de répercussion de l'état du scrolling.

#### **Tâche 16 : phase de tests**

- Date de début : 12/03/2021
- Date de fin : 17/03/2021
- Durée : 4 jours
- Description : cette tâche consiste à réaliser les tests fonctionnels et à écrire les tests unitaires.
- Livrable : application avec l'ensemble des fonctionnalités intégrées

#### **Tâche 17 : rédaction du rapport du semestre 10**

- Date de début : 19/03/2021
- Date de fin : 30/03/2021
- Durée : 8 jours
- Description : cette tâche consiste à rédiger le rapport du semestre 10.

#### **Tâche 18 : préparation à la soutenance du semestre 10**

- Date de début : 31/03/2021
- Date de fin : 08/04/2021
- Durée : 7 jours
- Description : cette tâche consiste à réaliser le support de présentation et à préparer la soutenance du semestre 10.

## **4 Gestion des risques**

Le principal risque identifié est celui lié à la réalisation de la tâche concernant le développement de la fonctionnalité d'ajout de formats de téléchargement du core. En effet, la réalisation de cette dernière nécessite l'utilisation de la technologie XSLT permettant la transformation d'un fichier au format XML dans un autre, que je ne maîtrise pas. De plus, cette tâche présente une complexité plus importante que les autres liée aux balises TEI présentes dans les fichiers XML. Afin de réduire la criticité de ce risque, j'alloue plus de temps pour cette tâche notamment pour me former à la technologie XSLT.

# CESR-DI : outil en ligne pour le balisage stand-off d'un appareil critique

Dylan MOTARD

Encadrement : Mohamed SLIMANE et Jean-Yves RAMEL



En collaboration avec Centre d'Études Supérieures de la Renaissance

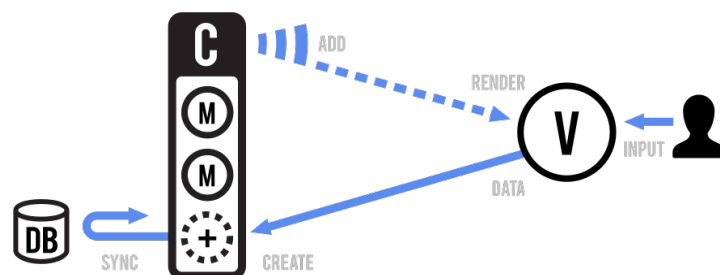
## Objectifs

- Ajouter des fonctionnalités à l'application Web existante
- Améliorer l'application Web existante



## Mise en œuvre

Le développement de l'application Web est réalisé à partir du framework JavaScript Backbone.js qui se base sur une architecture avec des modèles, des vues et des collections.



## Résultats attendus

L'objectif est d'obtenir un core qui est un fichier de sortie au format XML, HTML ou PDF contenant un ensemble de balises vides indiquant le lien entre les mots.

```
<app>
  <rdg wit="#E2">
    <ptr
      target="E2.xml#v1"/>
    </rdg>
    <rdgGrp>
      <rdg
        wit="#E2">
          <ptr
            target="S71.xml#v1"/>
          </rdg>
          <rdg
            wit="#E2">
              <ptr
                target="Trm0319a-Canto.xml#v1"/>
              </rdg>
            </rdgGrp>
          </rdg>
        </rdgGrp>
      </app>
```



# CESR-DI : outil en ligne pour le balisage stand-off d'un appareil critique

Dylan MOTARD

Encadrement : Mohamed SLIMANE et Jean-Yves RAMEL



En collaboration avec Centre d'Études  
Supérieures de la Renaissance

## Objectifs

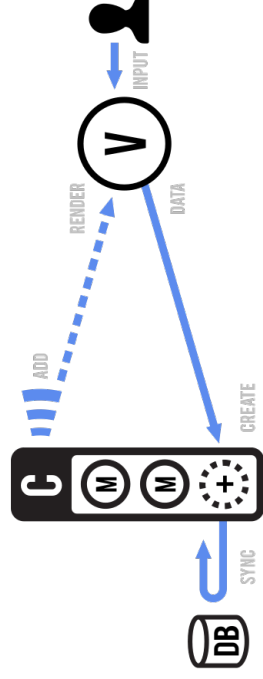
- Ajouter des fonctionnalités à l'application Web existante
- Améliorer l'application Web existante

## Mise en œuvre

Le développement de l'application Web est réalisé à partir du framework JavaScript Backbone.js qui se base sur une architecture avec des modèles, des vues et des collections.

## Résultats attendus

L'objectif est d'obtenir un core qui est un fichier de sortie au format XML, HTML ou PDF contenant un ensemble de balises vides indiquant le lien entre les mots.



```
<app>
  <rdg wit="#E2" >
    <ptr
      target="E2.xml#v1" />
    </rdg>
    <rdgGrp>
      <rdg
        wit="#E2" >
          <ptr
            target="S71.xml#v1" />
          </rdg>
          <rdg
            wit="#E2" >
              <ptr
                target="Trm0319a-Canto.xml#v1" />
              </rdg>
            </rdgGrp>
          </app>
```





# CESR-DI : outil en ligne pour le balisage stand-off d'un appareil critique

## Résumé

L'objectif du Projet de Recherche et de Développement "CESR-DI : outil en ligne pour le balisage stand-off d'un appareil critique" est de poursuivre le développement d'une application Web existante permettant de réaliser le balisage de documents textuels anciens en ajoutant des fonctionnalités à celle-ci.

## Mots-clés

Application Web, balisage, documents textuels anciens

## Abstract

The objective of the Research and Development Project "CESR-DI: online tool for the stand-off tagging of critical apparatus" is to continue the development of an existing web application that allows the tagging of old text documents by adding functionalities to it.

## Keywords

Web application, tagging, old text documents

## Entreprise

Centre d'Études Supérieures de la Renaissance



## Tuteur entreprise

Elena PIERAZZO

## Étudiant

Dylan MOTARD (DI5)

## Tuteurs académiques

Mohamed SLIMANE

Jean-Yves RAMEL