Université de Lille

MÉMOIRE NUMÉRIQUE

(MASTER 2)

Éditions numériques enrichies des romans *Allah n'est pas obligé* d'Ahmadou Kourouma et *Sozaboy (Pétit minitaire)* de Ken Saro-Wiwa

Mouhamadou Moustapha Deme

Mémoire soutenu le 15 août 2023

Devant le jury composé de :

Mme Marie Bulté – Maîtresse de conférences, Université de Lille 3, Directrice du Mémoire littéraire

M. Matthieu Marchal – Maître de conférences, Université de Lille 3, directeur du mémoire numérique

Année universitaire : 2022-2023

Remerciements

Tout d'abord je tiens à remercier mes directeur·trice·s de recherche, Mme Marie Bulté pour le mémoire littéraire et M. Marchal pour le mémoire numérique, d'avoir accepté de me suivre et d'avoir suivi de près ce travail de recherche, pour leurs soutiens ainsi que leurs conseils. Je remercie aussi mes parents qui ont toujours été là pour moi ainsi que mes proches, particulièrement, mon oncle Alhousseïni Niang et ma tante Mariame ly, mes grands frères Abdouhalimou Deme et Abdourahimou Deme. Un immense MERCI à Simon Deliege, dont la contribution a été indispensable pour la réussite de mon projet numérique. Mes pensées vont aussi à mes ami·e·s et connaissances qui m'ont apporté un soutien moral et intellectuel tout au long de ce travail en particulier : Pauline Modolo, Mélina Postel, Dounia Bendouma, Marie Malagnoux, Samba Diallo, Jean Antoine Dacosta, Ibrahima Ba.

Introduction

Ce mémoire numérique représente l'aboutissement d'une étude sur la représentation des enfants-soldats dans la littérature : le langage comme expression identitaire et de violence dans *Sozaboy (Pétit minitaire)* de Ken Saro-wiwa, *Allah n'est pas obligé* de Ahmadou Kourouma et *Bêtes sans patrie* de Uzodinma Iweala. L'objectif de ce mémoire numérique est de présenter une édition numérique enrichie des romans *Allah n'est pas obligé*[1] de Ahmadou Kourouma et *Sozaboy (pétit minitaire)*[2] de Ken Saro-Wiwa. Cette édition numérique vise tout public que ce soit.

Ces romans, bien que distincts dans leur style narratif, partagent une caractéristique commune qui a captivé notre intérêt : leur capacité à refléter la réalité de la guerre à travers leur écriture.

En ce qui concerne le roman *Allah n'est pas obligé*, nous avons pu analyser que la langue française est tropicalisée par le Malinké, dialecte que l'on retrouve dans le nord de la Côte-d'Ivoire. Cette hybridité langagière marquée par l'entrechoc entre le français et les langues vernaculaires pousse le lecteur à se soumettre à une traduction permanente tout au long de sa lecture. Ainsi dans Allah n'est pas obligé, nous avons un glossaire intradiégétique assumé par le narrateur enfant-soldat, Birahima, qui se permet de donner la signification de plusieurs termes lexicaux ou expressions. Le statut assumé ici par l'usage de 4 dictionnaires : « Ces dictionnaires me servent à chercher les gros mots, à vérifier les gros mots et surtout à les expliquer. Il faut expliquer parce que mon blablabla est à lire par toutes sortes de gens : des Toubabs (Toubab signifie blanc) colons, des noirs indigènes sauvages d'Afrique et des francophones de tout gabarit (gabarit signifie genre). [3]» Ahmadou Kourouma intègre sa glose au fil de la diégèse. Cette pratique permet d'alterner entre la narration des événements et les commentaires métalinguistiques. Il est bien vrai que la glose interlinéaire que l'on retrouve dans ce roman évite au lecteur exogène d'interrompre sa lecture. Néanmoins, le lecteur au fil de sa lecture peut ne plus se souvenir de la première définition d'un mot donnée par le narrateur ainsi que les références de la page où elle se trouve, par conséquent il sera obligé de reparcourir tout le texte. Notre objectif, sera dans ce projet est d'élaborer une édition enrichie à travers la création d'un

[1] Ahmadou Kourouma, *Allah n'est pas obligé*, Paris, Éditions du Seuil, 2000
[2] Ken Saro-Wiwa, *Sozaboy* [1985], *Sozaboy (Pétit minitaire)*, traduit de l' « anglais pourri » (Nigéria) par Samuel Millogo et Amadou Bissiri, Arles, Actes Sud, coll. « Babel », 2006.
[3] Ahmadou Kourouma, *Allah n'est pas obligé*, *op. cit.*, p. 9.

glossaire extra-diégétique de toutes les particularités lexicales du français en Afrique Noir et du dictionnaire Harrap's utilisé par Birahima et dans lequel nous aurons les liens hypertextes entre la glose dans le glossaire et le mot référencé dans le texte. Ceci permettra de faciliter la lecture du lecteur, tout en se retrouvant dans le texte.

Le roman *Sozaboy (pétit minitaire)* présente également des défis linguistiques à travers un langage distinctif qui reflète le conflit. Le récit du narrateur Méné, enfant-soldat, nous offre une expérience immersive et auditive en présentant la guerre à travers une narration principalement orale. C'est précisément cette dimension orale qui a motivé notre choix de réaliser une édition enrichie, comprenant une lecture audio d'un extrait du roman. Cette approche nous permettra de faire entendre la manière dont s'exprime un enfant-soldat, à travers une dislocation délibérée de la langue. Il est important de noter que cette édition audio sera basée sur la traduction de Samuel Millogo et d'Amadou Bissiri.

Au fil de ce projet, nous examinerons en détail les différentes étapes de sa réalisation, débutant par l'exploration du roman d'Ahmadou Kourouma avant de nous plonger dans celui de Ken Saro-Wiwa.

I. *Allah n'est pas obligé*

A. Récupération du texte au format ODT

La première étape a d'abord été de trouver le roman *Allah n'est pas obligé* dans une version numérique en document Word, PDF (Portable Document Format) ou Texte OpenDocument (ODT). Mais la préférence était de l'avoir en document ODT. Cette préférence est due au fait qu'un document .odt au format ODF[4] est en fait une archive zippée contenant un certain nombre de fichiers et de répertoires, ainsi qu'on peut le découvrir en l'ouvrant comme une archive Zip[5]. Le format OpenDocument soutient une forte séparation entre contenu, mise en page et métadonnées. Les fichiers sont représentés sous une arborescence en format XML[6], ce

[4] Le format ODF, ou Open Document Format, est un format de fichier ouvert et standardisé principalement utilisé pour les documents bureautiques.
[5] Une archive ZIP est un fichier compressé qui contient un ou plusieurs fichiers ou dossiers.
[6] L'XML, eXtensible Markup Language, est un langage informatique de balisage générique, c'est-à-dire qu'il s'écrit en utilisant des balises qui permettent de structurer et de hiérarchiser un document.

format sur lequel nous utiliserons pour les prochaines étapes de ce projet. Après avoir trouvé le roman en PDF sur le net, nous l'avons téléchargée et convertie en document ODT.

1. Stylage du texte

Après avoir converti le document, nous sommes passés à l'étape du stylage. Dans un éditeur de traitement de textes, comme Open Office (dans notre cas), nous disposons de fonctionnalités de styles permettant sous un seul label de regrouper tous les paramètres typographiques qui pourront être appliqués à des paragraphes ou à des portions de texte. Ainsi, nous distinguons deux principaux types de style :

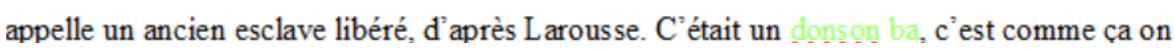
- Les styles de paragraphes qui concernent des paragraphes entiers ou un bloc de texte. Ils concernent les titres, les paragraphes.
- Les styles de caractères : une zone de texte en code informatique, les liens hypertextes, des mots ou un groupe de mots.

Ainsi, en stylant un texte, on applique des propriétés de mise en forme visuelle à différents éléments du document. Lorsqu'un texte est stylé en .odt, il est transformé en un format XML[7] où un balisage implicite est appliqué aux éléments stylés. Ce balisage, bien qu'invisible, nous sera essentiel pour la suite de notre travail.

Ce stylage donne déjà une structure implicite du futur document, car il définit des parties distinctes avec des caractéristiques visuelles spécifiques. Par exemple, en utilisant des styles de caractère pour les entrées, les gloses, les proverbes et les expressions dans notre cas, nous créons une structure claire dans le texte.

Dans notre contexte, le texte que nous avons récupéré en.odt avait déjà une structure identique à la structure de la version papier. Donc il n'était pas assez nécessaire d'appliquer des styles de paragraphes. Nous nous sommes juste limités aux styles de caractère. Les styles qui ont été utilisés pour appliquer une mise en évidence à certaines parties du texte sont les suivantes :

- L'entrée principale d'une glose a été stylée avec une couleur verte avec comme nom : « ent ».


appelle un ancien esclave libéré, d'après Larousse. C'était un donson ba, c'est comme ça on

[7] L'XML, eXtensible Markup Language, est un langage informatique de balisage générique, c'est-à-dire qu'il s'écrit en utilisant des balises qui permettent de structurer et de hiérarchiser un document. Pour en savoir plus sur l'XML : https://openclassrooms.com/courses/structurez-vos-donnees-avec-xml/qu-est-ce-que-le-xml

- Les gloses ont été stylées avec une couleur rouge avec comme nom : « def ».

appelle un maître chasseur qui a déjà tué un fauve noir et un génie malfaisant, d'après Inventaire

- Les proverbes ont été stylés avec une couleur bleue avec comme nom : « prvb ».

Suis dix ou douze ans (il y a deux ans grand-mère disait huit et maman dix) et je parle beaucoup. Un enfant poli écoute, ne garde pas la palabre… Il ne cause pas comme un oiseau gendarme dans les branches de figuier. Ça, c'est pour les vieux aux barbes abondantes et blanches, c'est ce que dit le proverbe : le genou ne porte jamais le chapeau quand la tête est sur le cou. C'est ça les coutumes au village. Mais moi depuis longtemps je m'en fous des coutumes du village, entendu que j'ai été au Liberia, que j'ai tué beaucoup de gens avec kalachnikov (ou kalach) et me suis bien camé avec kanif et les autres drogues dures.

- Les expressions ont été stylées avec une couleur violette avec comme nom : « expr ».

ça peut te mettre une abeille vivante dans ton œil ouvert.

B. Étapes de la conversion du fichier ODT en XML

1. Convertir le fichier ODT en un fichier XML

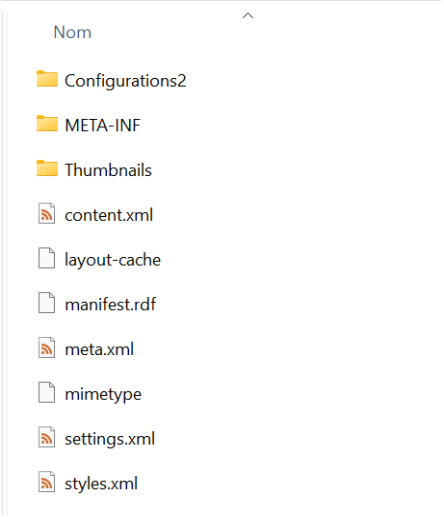
Comme mentionné précédemment, le fichier ODT est une archive complexe qui renferme divers dossiers et fichiers. Lorsqu'un texte est stylé en appliquant des propriétés de mise en forme visuelle, il dénote implicitement une structure dans le document. Lors de la conversion vers un format XML, cette structure est ensuite rendue explicite au moyen d'une balise sémantique. Pour extraire le fichier XML nécessaire à la poursuite du travail, plusieurs étapes sont nécessaires.

2. Méthodologie de traitement ODT vers XML

- **Stylage Complet** : Tout d'abord, le document ODT est soumis à un processus de stylage complet dans un logiciel de traitement de texte. Cela implique l'application cohérente de styles de paragraphe et de caractères pour formater le texte conformément aux besoins.

- **Création d'une Copie au Format ZIP** : Une copie du fichier ODT est créée afin de ne pas perdre les données du fichier principal, puis l'extension de cette copie est modifiée en .zip, la transformant ainsi en une archive ZIP distinct portant le nom "Stylage_Allah_n'est_pas_oblige-copie.zip."

- **Décompression du Fichier ZIP** : Cette archive ZIP est ensuite extraite dans un dossier portant le même nom que l'archive (c'est-à-dire "Stylage_Allah_n'est_pas_oblige-copie"). Dans ce dossier, une variété de fichiers XML et de dossiers est découverte.



Exemple d'un dossier dézippé issu d'un fichier .odt

Le fichier "styles.xml" contient des définitions pour tous les styles utilisés dans le document, tandis que le fichier "content.xml" contient le contenu réel du document au format XML. Ce contenu comprend essentiellement le texte saisi dans le document, à l'exception des éléments binaires tels que les images, qui sont stockés dans des fichiers séparés. Le fichier "content.xml" revêt une importance particulière, car c'est le fichier que nous utiliserons principalement.

- **- Édition avec un Éditeur XML** : Le fichier "content.xml" est ouvert dans un éditeur XML approprié, tel qu'Oxygen editor. Le fichier "content.xml" est une représentation brute (annexe I) où l'on trouve des informations de base sur le document, les styles appliqués, ainsi que des commandes XML. Pour éliminer les informations superflues, notamment les noms de styles, un fichier XSLT (Extensible Stylesheet Language Transformations) est utilisé. Le XSLT est un langage de programmation permettant de formater les données XML dans divers formats (par exemple, XML vers XHTML[8],

[8] L'XHTML (eXtensible HyperText Markup Language) est un langage de balisage utilisé pour structurer et formater le contenu des pages web. Il est basé sur XML (eXtensible Markup Language) et suit les règles strictes de la syntaxe XML.il est considéré comme une évolution du HTML.

HTML[9], CSV[10], texte, PDF, etc.). Dans notre cas, la transformation vise à convertir un fichier XML en un autre fichier XML. Le fichier XSLT utilisé à cet effet, nommé "ODTversXML=Audrain+Gasiglia_ODT2017-02.xslt,» nous a été fourni par Madame Gasiglia lors d'un devoir en première année. Les balises[11] supprimées sont : **text:s**, **office:scripts**, **office:forms**. Les balises englobantes telles que **office:document-content**, **office:body**, et **office:text** sont également supprimées ; les éléments **text : p** (paragraphe) sont transformés en balises dont le nom dépend des styles appliqués. Les éléments **text:span** (portion de texte ou caractères) sont également transformés en balises en fonction des styles appliqués, avec des règles de correspondance similaires. Le fichier généré est nommé : "Allah_n_est_pas_oblige.xml."

- **Suppression d'autres éléments parasites** : Pour garantir l'élimination de tout autre élément non souhaité, une étape ultérieure consiste à appliquer au fichier XML un fichier XSLT supplémentaire nommé "XML_vers_XML = suppr_Ti.xsl pour supprimer les noms des attributs des styles. Ce fichier nous a toujours été fourni par Madame Gasiglia. Cette étape permet de finaliser le fichier XML résultant toujours nommé Allah_n_est_pas_oblige.xml. Ce fichier ne contient plus les éléments indésirables et inutiles du document ODT d'origine. (Annexe II)

Le fichier XML final conserve la structure du document initial qui avait été stylé. Les styles de paragraphes utilisés dans le traitement de texte correspondent dans la structure XML à la notion de "bloc". Les styles de caractères, quant à eux, correspondent à la notion de "textinline" dans l'XML. Les noms des styles des entrées, les gloses et les proverbes sont désormais représentés par des balises XML appropriées, telles que <ent>, <def>, <prvb>, <expr>, etc. à l'intérieur duquel nous retrouvons le texte mis en valeur.

En somme, ces étapes permettent de convertir le fichier ODT en un fichier XML structuré en fonction des noms de styles utilisés dans le traitement de texte, tout en éliminant les éléments superflus. Cette approche facilite grandement la manipulation et l'utilisation ultérieure du contenu dans des langages informatiques tels que XML et HTML.

---

[9] Le HTML (HyperText Markup Language) est le langage de balisage standard utilisé pour créer et structurer le contenu des pages web. Il est composé d'éléments (balises) qui définissent la structure et la mise en forme du contenu, tels que les titres, les paragraphes, les images, les liens, etc.

[10] Le CSV (Comma-Separated Values) est un format de fichier couramment utilisé pour stocker et échanger des données tabulaires, telles que des feuilles de calcul.

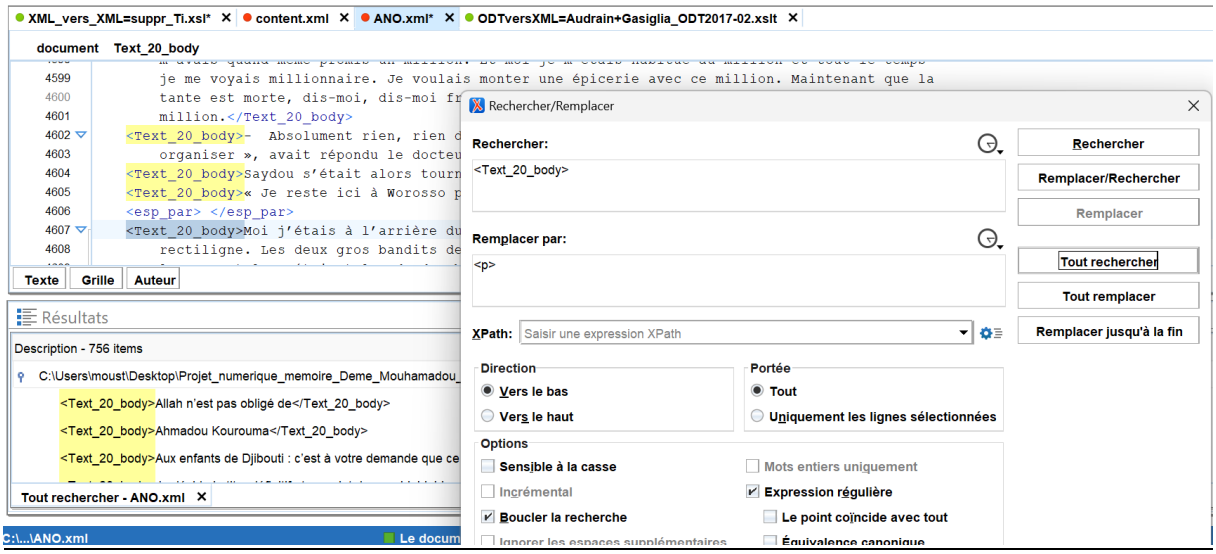
[11] Les balises sont les composants fondamentaux permettant l'écriture de documents XML. Elles se distinguent du contenu en utilisant les caractères <, > et /. Elles n'ont pas de présentation ou de signification définie par le langage, mais elles auront un sens pour les applications et/ou le lecteur. La syntaxe d'une balise est représentée ainsi : Il existe trois types de balises : balise d'ouverture : <nom_balise> ; balise de fermeture : </nom_balise> balise vide : <nom_balise/>

3. Création d'un autre fichier XML vers XML par transformation XSLT

Lors du processus de stylage, nous n'avions décidé de ne pas appliquer de style à tous les paragraphes, retraits entre les paragraphes et dialogues, car la structure était claire. Mais après avoir généré le fichier XML épuré, nous avons constaté que les noms des balises qui les représentaient n'étaient pas assez distincts, car ils avaient tous les mêmes noms de balise : <Text_20_body>. Par conséquent, nous avions effectué une étape de recherche et remplacement pour modifier les noms des balises afin de rendre la structure du document plus cohérente, j'ai remplacé ces balises "<Text_20_body>" par des balises plus appropriées. Par exemple, j'ai utilisé la balise "<p>" pour les blocs de paragraphes, "<esp_par>" pour les retraits entre les paragraphes, et "<said>" pour les dialogues. Dans la même logique, nous avons décidé de générer un identifiant unique pour chaque bloc de paragraphe, car nous avions pensé que cela nous serait utile pour une meilleure navigation entre le texte et le glossaire. Pour le faire automatiquement nous avons généré un fichier .xsl qui s'applique à notre fichier XML de base. Le fichier .xsl a été nommé : renommation_des_balises_en_TEI_plus_integration_id.xsl

Cette étape de recherche et remplacement avait pour objectif de bien baliser le document et nous permettre une meilleure manipulation et une analyse ultérieure plus faciles du texte en utilisant des balises spécifiques pour chaque élément structurel du document.

"<p>"

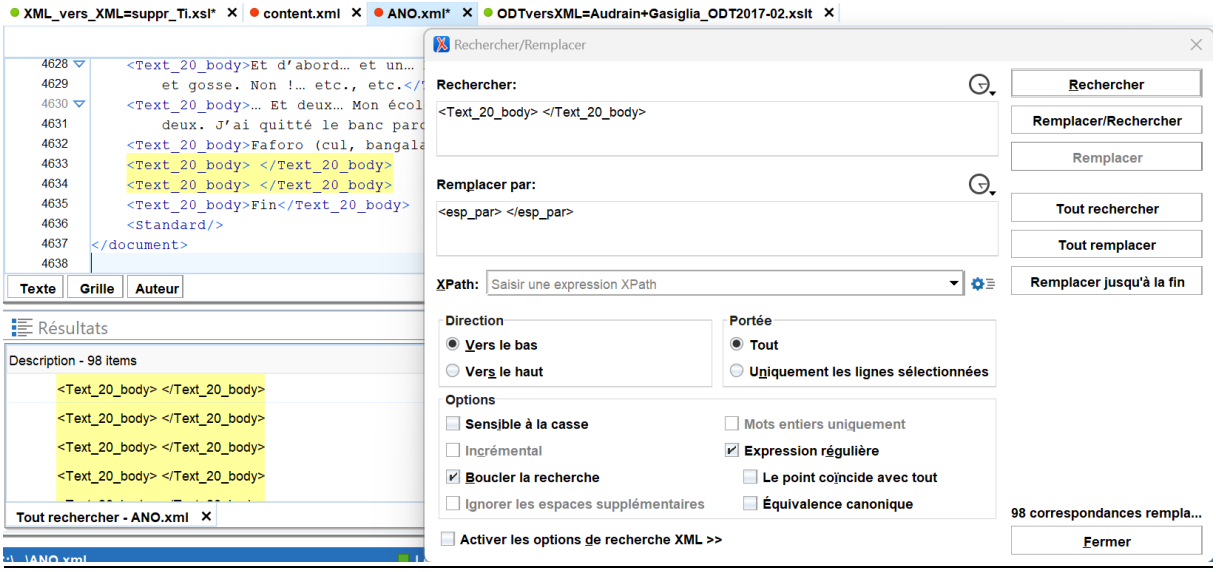


```xml
<xsl:template match="p">
    <!-- Génération de l'identifiant unique pour chaque paragraphe -->
    <xsl:variable name="id" select="generate-id()"/>

    <!-- Copie de la balise de paragraphe avec l'attribut "id" -->
    <xsl:copy>
        <xsl:attribute name="id">
            <xsl:value-of select="$id"/>
        </xsl:attribute>

        <!-- Copie du contenu de la balise de paragraphe -->
        <xsl:apply-templates select="@*|node()"/>
    </xsl:copy>
</xsl:template>
```

À l'aide de l'outil rechercher/remplacer nous avons remplacé certaines balises <Text_20_body> par une balise <p>. Dans le fichier xsl, on génère un identifiant unique pour chaque paragraphe.

"<esp_par>"

À l'aide de l'outil rechercher/remplacer nous avons remplacé certaines balises <Text_20_body> par une balise <esp_par>.



<Said>

À l'aide de l'outil rechercher/remplacer nous avons remplacé certaines balises <Text_20_body> par une balise <said>.

Toujours dans un souci de lisibilité et de permettre une meilleure manipulation du fichier XML, nous avions décidé de renommer les noms des balises : <ent> par <term> ; <expr> par <q> ; <prvb> par <q> ; et <def> par <gloss> et parallèlement, nous avions généré des identifiants uniques pour chaque balise pour un éventuel lien entre le fichier glossaire.html et le fichier HTML de la version numérique du glossaire. Les noms de balises nous sont venus des nominations d'éléments TEI (Text Encoding Initiative)[12]. Nous aurions pu procéder par des chercher-remplacer pour le faire comme dans les étapes précédentes, mais pour générer des identifiants automatiquement pour chaque balise, il fallait le faire par un programme. Alors nous avions rajouté des codes dans le fichier renommation_des_balises_en_TEI_plus_integration_id.xsl.

- <ent>

```
<xsl:template match="ent"> <!-- remplacement des balises ent en term, avec un rajout d'identifiant correspondant au nom de l'entrée -->
    <xsl:element name="term">
        <xsl:attribute name="id">
            <!--<xsl:value-of select="upper-case(translate(., 'àâêéèîïôùçÀÂÊÉÈÎÏÔÙÇ ','aaeeeeiioucAAAEEEIIOUC'))"/>-->
            <xsl:value-of select="."/>
        </xsl:attribute>
        <xsl:apply-templates/>
    </xsl:element>
</xsl:template>
```

- <q>

```
<xsl:template match="expr">
    <xsl:element name="q">
        <xsl:attribute name="ana">
            <xsl:attribute name="nom">expr</xsl:attribute>
        </xsl:attribute>
        <xsl:apply-templates/>
    </xsl:element>
</xsl:template>
```

- <gloss>

[12] La Text Encoding Initiative (TEI) est une initiative internationale fournissant des directives de codage XML pour la représentation numérique avancée de textes littéraires et linguistiques. Elle permet la structuration, l'analyse et la publication électronique de documents textuels.

```
<xsl:template match="def"><!-- remplacement des balises def en gloss, avec un rajout d'identifiant correspondant au nom de la glose -->
    <xsl:element name="gloss">
        <xsl:attribute name="id">
            <xsl:value-of select="preceding-sibling::ent[1]"/>
            <xsl:value-of select="preceding-sibling::expr[1]"/>
        </xsl:attribute>
        <xsl:apply-templates/>
    </xsl:element>
</xsl:template>
```

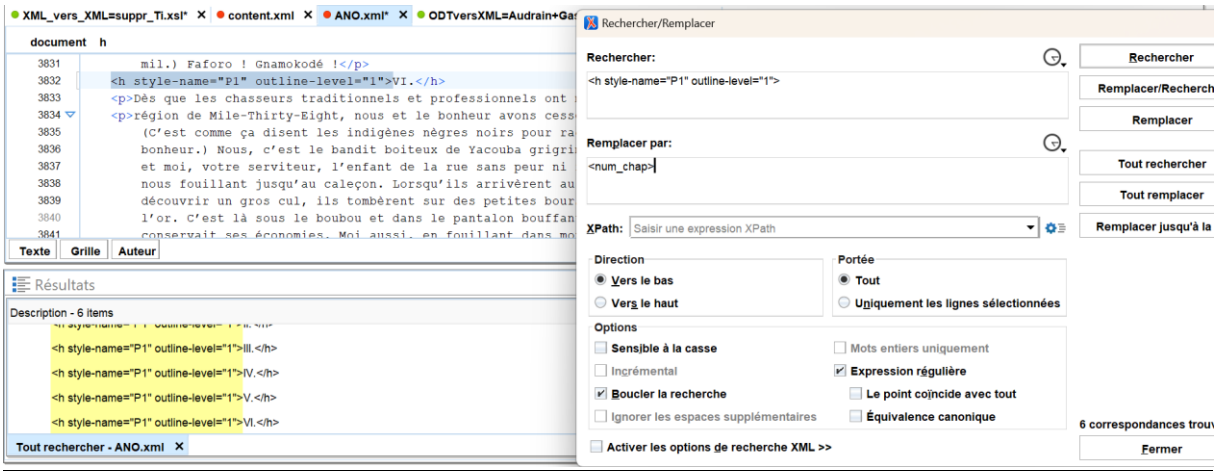
➢ Difficultés

Lors de la génération des identifiants des gloses correspondant au contenu de l'élément, les retours chariot générés lors de la transformation du content.xml étaient pris en compte. Par exemple, nous avions ce résultat lorsque les phrases des contenus d'éléments étaient longues : <gloss id="de
 l'initiation ">considérer comme un vrai copain</gloss>. Par conséquent, il a fallu que nous créions un nouveau fichier XSL (avec comme sortie un fichier XML) qui nous permettrait de supprimer tous les retours chariot.

```
    <xsl:output encoding="UTF-8" method="xml" indent="yes"/>
    <xsl:template match="text()">
        <xsl:value-of select="translate(., '&#xD;', '')"/>
        <xsl:apply-templates/>
    </xsl:template>

    <!-- Copie les autres éléments et attributs inchangés -->
    <xsl:template match="@* | node()">
        <xsl:copy>
            <xsl:apply-templates select="@* | node()"/>
        </xsl:copy>
    </xsl:template>
</xsl:stylesheet>
```

Num

À l'aide de l'outil rechercher/remplacer nous avons remplacé les balises <h text:style-name="P1" text:outline-level="1VI.</h> par une balise <num_chap>.




```xml
<xsl:template match="num_chap"><!-- remplacement des balises num_chap en num -->
    <xsl:element name="num">
        <xsl:apply-templates/>
    </xsl:element>
</xsl:template>
```

C. LES ÉTAPES DE TRANSFORMATION VERS HTML

1. Élaboration d'un document HTML pour l'œuvre *Allah n'est pas obligé* avec une feuille de styles CSS[13].

Dans cette partie du projet, j'ai développé un code XSLT pour effectuer une transformation du fichier Allah_n_est_pas_obligé.xml en un document HTML. Cette transformation visait à créer une représentation web de l'œuvre. Le processus de transformation impliquait les étapes suivantes :

Ajout d'ancres et de liens hypertextes sur les balises <term>, <gloss> et <q>. Ces liens permettent à l'utilisateur de naviguer vers le glossaire si besoin. Dans ce fichier XSLT, nous avons ajouté des ancres et des liens hypertextes sur les balises des termes, des gloses, des proverbes et des expressions, pour une meilleure navigation bidirectionnelle entre les entrées du texte principal et les entrées du glossaire qui sera généré dans la suite de notre projet.

Plus précisément, pour chaque élément **<q>** ou **<term> …** dans le fichier XML source, nous avons utilisé un code XSLT pour créer des liens hypertextes avec des attributs **class**, **href**, et **id**. L'attribut **class** a été défini comme "term" pour appliquer des styles CSS spécifiques aux

[13] La CSS, ou feuille de style en cascade (Cascading Style Sheet en anglais), est un langage utilisé pour définir la présentation d'un document HTML. La CSS permet de spécifier des règles de mise en forme, de disposition et d'apparence des éléments HTML.

liens. L'attribut **href** a été configuré pour pointer vers le fichier HTML "Allah_n_est_pas_oblige.html", en utilisant le contenu de l'élément comme ancre pour naviguer vers des sections spécifiques.

- Le document HTML obtenu a été lié à une feuille de style CSS.

En reliant le document HTML de l'œuvre *Allah n'est pas obligé* à une feuille de style CSS, nous avons pu définir les polices, les couleurs, les marges, les alignements, et d'autres aspects visuels pour rendre le contenu HTML esthétiquement attrayant et bien structuré.

D. Création d'un fichier HTML pour le glossaire.

1. Objectif

Dans cette partie, L'objectif au départ était de mettre en avant le roman de Kourouma, *Allah n'est pas obligé,* mais aussi sur *Sozaboy (Pétit minitaire)* de Ken Saro-Wiwa (voir remarque). Mais par manque de temps nous avons choisi de nous focaliser sur *Allah n'est pas obligé.*

Dans *Allah n'est pas obligé*, nous observons que la langue française est tropicalisée par le Malinké, dialecte que l'on retrouve dans le nord de la Côte-d'Ivoire. Cette hybridité langagière marquée par l'entrechoc entre la langue standard qui est le français et les langues vernaculaires pousse le lecteur à se soumettre à une traduction permanente tout au long de sa lecture. Ainsi dans *Allah n'est pas obligé*, nous avons un glossaire intradiégétique assumé par le narrateur enfant-soldat, Birahima, qui se permet de donner la signification de plusieurs termes lexicaux africains ou expressions. Le statut assumé ici par l'usage de 4 dictionnaires : « Ces dictionnaires me servent à chercher les gros mots, à vérifier les gros mots et surtout à les expliquer. Il faut expliquer parce que mon blablabla est à lire par toutes sortes de gens : des Toubabs (Toubab signifie blanc) colons, des noirs indigènes sauvages d'Afrique et des francophones de tout gabarit (gabarit signifie genre).[14] ».

Ahmadou Kourouma intègre sa glose au fil de la diégèse. Cette pratique permet d'alterner entre la narration des événements et les commentaires métalinguistiques. Il est bien vrai que la glose interlinéaire que l'on retrouve dans ce roman évite au lecteur exogène d'interrompre sa lecture. Néanmoins cette pratique peut être contraignante dans la mesure où le lecteur au fil de sa lecture peut ne plus se souvenir de la première définition d'un mot donnée par le narrateur

[14] Ahmadou Kourouma, *Allah n'est pas obligé*, *op. cit.*, p. 9.

ainsi que les références de la page où elle se trouve, par conséquent il sera obligé de reparcourir tout le texte.

Notre objectif a été dans ce projet :

- D'élaborer un glossaire extradiégétique de toutes les particularités lexicales du français en Afrique Noir et du dictionnaire Harrap's défini par Birahima, dans lequel nous aurons les liens hypertextes entre la glose dans le glossaire et le mot référencé dans le texte (de faire un va-et-vient entre le texte et le glossaire). Ceci permettra de faciliter la lecture du lecteur, tout en se retrouvant dans le texte. Lors du balisage des points à mettre en évidence, nous avions décidé de ne pas rajouter de nouvelle entrée qui n'était pas définie par le narrateur. Nous aurions aimé classer notre glossaire par ordre alphabétique pour un meilleur repérage, mais nous nous sommes confrontés à une difficulté due aux identifiants de chaque paragraphe. Lorsqu'on essaye de les classer par ordre, les identifiants de chaque paragraphe changeaient et ne correspondaient plus aux identifiants générés dans le texte principal.

2. Création d'un code XSLT qui génère un fichier glossaire.xml

Pour pouvoir générer un glossaire extradiégétique dans une version numérique, nous avions besoin du fichier Allah_n_est_pas_obligé.xml où nous retrouvons tous les éléments liés à un glossaire. Pour le faire, nous avons créé un fichier XSLT qui nous permet d'extraire les balises <p> contenant les balises <term>, <gloss> et <q> à partir du fichier "Allah_n_est_pas_obligé.xml ". Les éléments extraits sont enregistrés dans un fichier XML nommé : result_extraction_des-term_gloss_q.xml

Dans cette transformation, j'ai également choisi de conserver les identifiants des balises <p> du fichier "Allah_n_est_pas_obligé.xml" au cas où ils seraient nécessaires. Ces identifiants sont ainsi préservés dans le fichier "glossaire.xml" pour une éventuelle référence ultérieure.

➢ Difficultés rencontrées

Lors du premier essai de génération du fichier XML, nous avons rencontré des problèmes avec les paragraphes <p>. Certains paragraphes contenaient deux entrées et deux gloses, tandis que d'autres paragraphes présentaient seulement une entrée suivie de sa glose. Cela était dû à des cas où nous avions plusieurs entrées dans un seul paragraphe lors du stylage de texte, ce qui ne correspondait pas à notre intention d'avoir un seul bloc <p> contenant une seule entrée suivie de sa glose.

Pour résoudre ce problème, nous avons modifié le code en subdivisant certains <p>. Ainsi, chaque subdivision contient uniquement un terme et sa glose, tout en conservant le même identifiant pour chacune des subdivisions.

```
<p id="d2e131">
   <term>cafre</term>
   <gloss>un homme qui refuse
    la religion musulmane et qui est plein de fétiches</gloss>
   <term>koroté</term>
   <gloss>poison opérant à distance sur la personne visée</gloss>
</p>
```

```
<p id="d2e131">
   <term>cafre</term>
   <gloss>un homme qui refuse
    la religion musulmane et qui est plein de fétiches</gloss>
</p>
<p id="d2e211">
   <term>koroté</term>
   <gloss>poison opérant à distance sur la personne visée</gloss>
</p>
```

3. Génération d'un fichier "glossaire.html" à partir du fichier "glossaire.xml"

En utilisant le fichier « glossaire.xml » comme source, nous avons créé un autre fichier XSLT qui a permis de générer un fichier « glossaire.html ». Dans ce fichier XSLT, nous avons ajouté des ancres et des liens hypertextes sur les balises des termes, des gloses, des proverbes et des expressions. Cela permet de naviguer vers le texte original.

Plus précisément, pour chaque élément **<q>** ou **<term>** dans le fichier XML source, nous avons utilisé un code XSLT pour créer des liens hypertextes avec des attributs **class**, **href**, et **id**. L'attribut **class** a été défini comme "term" pour appliquer des styles CSS spécifiques aux liens. L'attribut **href** a été configuré pour pointer vers le fichier HTML "Allah_n_est_pas_oblige.html", en utilisant le contenu de l'élément comme ancre pour naviguer vers des sections spécifiques.

De plus, nous avons reproduit cette fonctionnalité dans le fichier « Allah_n_est_pas_oblige.html », permettant ainsi des allers-retours faciles entre le glossaire et le texte principal, offrant une expérience de navigation bidirectionnelle.

Enfin, nous avons relié le fichier glossaire.html à une feuille de styles CSS. Cette feuille de styles a été utilisée pour définir la mise en page et l'apparence visuelle du fichier HTML, en spécifiant des règles de présentation telles que les polices, les couleurs, les marges, les alignements, etc.

II. *Sozaboy (pétit minitaire)*

A. Génération d'un fichier HTML : sozaboy.html

1. Objectif

Lorsque nous nous plongeons dans la lecture du roman *Sozaboy (pétit minitaire)*, nous remarquons un univers marqué par la guerre où les langues se trouvent confronter à des défis particuliers. En effet, ce récit décrit la guerre de manière audacieuse à travers une déconstruction des langues standards, mettant en évidence un conflit linguistique qui se manifeste par un écart par rapport aux règles établies.

Parallèlement, *Sozaboy (pétit minitaire)* offre une expérience auditive unique en permettant de percevoir la guerre à travers un discours principalement oral. Le témoignage du narrateur contribue à créer un espace immersif, offrant une représentation plus vivante de la guerre.

Cet aspect oral est l'une des raisons principales qui nous ont poussés à intégrer une partie de notre projet sous la forme d'un livre audio au format HTML. L'objectif de cette section est de créer un fichier HTML incluant un bouton permettant la lecture audio de certains passages du roman *Sozaboy (pétit minitaire)*. Le texte écrit serait mis en évidence en synchronisation avec la lecture audio.

.

2. Étapes

*a) **Récupération du texte par océrisation :***

Face à l'indisponibilité du roman *Sozaboy (Petit militaire)* au format numérique, nous avons dû opter pour une méthode alternative de récupération du texte. Pour ce faire, nous avons décidé de prendre des photographies des passages spécifiques du roman à partir de sa version papier. Ces passages photographiés étaient ceux qui nous étaient nécessaires pour notre travail.

Une fois les photographies prises, nous avons utilisé une technique de reconnaissance optique de caractères (OCR) pour extraire le texte contenu dans ces images. L'OCR est un processus automatisé qui analyse les images des textes imprimés, identifie les caractères et les convertit en texte électronique éditable. Cette approche nous a permis de transformer les informations du roman papier en un format numérique que nous pouvions manipuler et utiliser plus facilement, facilitant ainsi notre accès aux données dont nous avions besoin.

*b) **Création du fichier HTML :***

Après l'étape d'OCR, nous avons entrepris la création d'un fichier HTML spécialement dédié à l'intégration du texte du roman. Contrairement à notre travail précédent avec le texte d'Allah n'est pas obligé, dans ce cas-ci, il n'était pas nécessaire d'appliquer des styles spécifiques au texte, puisque nous n'avions pas d'éléments à mettre en évidence ou à formater de manière complexe. Au lieu de cela, la mise en forme du texte pouvait être réalisée directement à travers le code HTML, simplifiant ainsi le processus de présentation du contenu du roman dans un format lisible et adapté à nos besoins.

*c) **Enregistrement audio :***

À la suite de cela, nous avons procédé à l'enregistrement audio du texte, créant ainsi un fichier au format MP3. Cette démarche avait pour but de fournir une version audio du texte, permettant ainsi une lecture synchronisée.

*d) **Création d'un fichier JavaScript[15] :***

Un code JavaScript a été développé et lié au fichier sozaboy.html. Ce code JavaScript permet de surligner le texte en synchronisation avec la lecture audio. Les principales étapes du code ont été commentées à l'intérieur du fichier JavaScript.

Le JavaScript a été utilisé dans cette partie pour ajouter une fonctionnalité dynamique à la page HTML. Voici quelques raisons pour lesquelles le JavaScript a été utilisé :

- Interaction utilisateur : Le JavaScript permet de gérer les interactions de l'utilisateur avec la page. Par exemple, en cliquant sur les boutons de démarrage, d'arrêt et de réinitialisation, l'utilisateur peut contrôler la lecture audio et le minuteur.

[15] Le JavaScript ou JS est un langage de programmation essentiel pour le développement de sites web interactifs, offrant des fonctionnalités dynamiques visant à améliorer l'expérience des utilisateurs sur ces sites.

- Manipulation du DOM : Le JavaScript peut être utilisé pour manipuler le Document Object Model (DOM) de la page HTML. La manipulation du DOM est l'une des principales utilisations de JavaScript dans le contexte du développement web. Le DOM est une représentation hiérarchique de la structure d'une page HTML, où chaque élément HTML (comme des paragraphes, des titres, des images, des liens, des div, etc.) est représenté comme un objet dans un arbre. Dans notre contexte, il est utilisé pour accéder aux éléments HTML, tels que les paragraphes ou les div contenant les phrases du texte, et pour changer leur style en surlignant les phrases en cours de lecture.

- Gestion du temps : Le JavaScript est idéal pour gérer le temps dans une application web. Il est utilisé ici pour mettre à jour le minuteur chaque seconde et vérifier si une phrase doit être surlignée à un moment spécifique dans la lecture audio.

- Intégration avec l'audio : Le JavaScript permet d'interagir avec l'élément audio HTML. Il est utilisé pour démarrer, arrêter et réinitialiser la lecture audio, ainsi que pour suivre le temps écoulé.

En utilisant JavaScript, la lecture audio et le surlignage synchronisé du texte sont rendus possibles, offrant une expérience interactive aux utilisateurs du fichier sozaboy.html.

Lors de cette transformation, nous avions rencontré différentes difficultés

e) Mise en évidence des lignes du texte

L'objectif initial était de mettre en évidence les lignes du texte en fonction des pauses naturelles, telles que les virgules, les points ou les points d'interrogation. Cependant, cela aurait demandé un effort considérable en termes de temps, car il aurait fallu écouter attentivement l'audio, noter précisément les durées de chaque partie et coordonner la mise en évidence en fonction de ces durées. En raison des contraintes de temps, la décision a été prise de simplifier le processus de la mise en évidence en ajoutant des identifiants pour chaque partie de paragraphe. Heureusement, l'écriture du roman *Sozaboy* a facilité cette tâche en offrant des phrases longues et moins de ponctuation.

f) Coordination de l'audio, de la mise en pause et de la mise en évidence

Une difficulté supplémentaire a été rencontrée lors de la mise en pause de l'audio. Lorsque la lecture audio était en pause, la mise en évidence du texte continuait à se dérouler, ce qui était contraire à l'objectif souhaité. La difficulté résidait dans la coordination entre l'audio, la mise en pause et la mise en évidence. Il était nécessaire de trouver un moyen de faire en sorte que la mise en évidence s'arrête lorsque l'audio est en pause et qu'elle reprenne lorsque l'audio est relancé. Cela a nécessité une réflexion et une mise en place de la logique appropriée dans le code JavaScript pour gérer cette coordination.

Ainsi, nous avons rajouté des gestionnaires d'événements JavaScript pour les boutons de contrôle (Start, Stop, Reset). Un gestionnaire d'événements est une fonction ou un morceau de code JavaScript spécialement conçu pour réagir à des événements déclenchés par un utilisateur ou par le navigateur web. Les événements sont des actions ou des occurrences qui se produisent dans une page web, comme un clic de souris, une pression sur une touche du clavier, le chargement de la page, le déplacement de la souris, etc. les gestionnaires d'événements qui ont été rajoutées sont :

start.onclick : Ce gestionnaire d'événements est activé lorsque l'utilisateur clique sur le bouton « Start ». Il commence la lecture de l'élément audio et démarre le compteur de temps (**timer()**), qui est responsable de la synchronisation des éléments HTML avec la lecture audio.

stop.onclick : Lorsque l'utilisateur clique sur le bouton « Stop », ce gestionnaire d'événements est déclenché. Il met en pause la lecture audio et annule le décompte du temps en cours en appelant **clearTimeout(t)** qui est une fonction JavaScript qui permettant d'annuler l'exécution d'une minuterie (ou "timeout"). Cela arrête la mise en évidence des phrases synchronisées avec l'audio.

reset.onclick : Ce gestionnaire d'événements est lié au bouton "Reset". Lorsque l'utilisateur clique sur ce bouton, il effectue plusieurs actions :

- Il annule le décompte du temps en cours (**clearTimeout(t)**) pour arrêter le chronomètre.
- Il désactive la mise en évidence de la phrase synchronisée avec l'audio
- Il réinitialise l'affichage de l'horloge à "00:00:00" (**h1.textContent**).
- Il réinitialise les variables **sec**, **min**, et **hrs** à zéro pour remettre le chronomètre à zéro.
- Il recharge l'élément audio (**audio.load()**) pour remettre la lecture audio au début.

- Enfin, il démarre à nouveau le compteur de temps (**timer()**) pour permettre une nouvelle synchronisation avec l'audio.

Ces gestionnaires d'événements permettent à l'utilisateur de contrôler la lecture audio, de mettre en pause la synchronisation des éléments HTML avec l'audio, et de réinitialiser le chronomètre et la synchronisation au début. Ils contribuent à offrir une expérience d'écoute interactive avec des fonctionnalités de contrôle.

Ces difficultés ont été des défis à surmonter lors de la réalisation du travail, et des compromis ont été faits pour adapter les fonctionnalités en fonction des contraintes de temps et de complexité. Malgré ces obstacles, le résultat final a permis de fournir une expérience interactive avec la lecture audio et le surlignage synchronisé du texte.

III. Page d'accueil

Dans cette partie, nous avons généré un fichier HTML appelé index.html qui servira de page d'accueil avec un menu de navigation. Ce menu comprendra des sous-menus déroulants pour faciliter l'accès aux différentes sections du site.

A. Sous-menu Texte

Ce sous-menu contiendra deux parties :

- *Allah n'est pas obligé :* cette partie nous redirigera vers le fichier Allah_n_est_pas_obligé.html qui contient le texte du roman *Allah n'est pas obligé*.
- *Sozaboy* : Cette partie redirigera vers le fichier sozaboy.html qui contient le texte du roman *Sozaboy*.

B. Sous-menu Répertoire lexicographique

Ce sous-menu contiendra des liens vers le glossaire du texte *Allah n'est pas obligé*.

L'objectif de ce fichier index.html est de fournir une interface utilisateur conviviale avec un menu de navigation clair pour accéder facilement aux différentes parties du site, notamment les textes *Allah n'est pas obligé* et *Sozaboy*, ainsi que le répertoire lexicographique.

Pour faciliter la navigation entre les pages, nous allons insérer aussi ce menu sur toutes les pages web.

Conclusion

Pour conclure, ce projet d'édition numérique enrichie des romans *Allah n'est pas obligé* et *Sozaboy (pétit minitaire)* avait pour premier objectif de mettre en avant les thèmes explorés dans ma thèse littéraire, en particulier ceux traités dans la deuxième partie concernant l'écriture de la violence et la crise du langage. En parallèle, il a été l'occasion de démontrer les compétences numériques que nous avions acquises au cours de ces deux années de master.

Bien que nous n'ayons pas pu réaliser intégralement notre projet initial, notamment en ce qui concerne l'édition bilingue de "Sozaboy," qui aurait mis en évidence notre travail sur les défis et les stratégies de traduction, nous sommes parvenus à concrétiser la plupart des idées initiales. Nous avons créé une édition numérique enrichie du roman *Allah n'est pas obligé* en utilisant un glossaire extradiégétique visant à améliorer la compréhension pour les lecteurs.

De plus, nous avons développé une édition enrichie d'un extrait du roman *Sozaboy (pétit minitaire)*, en intégrant une lecture audio pour permettre aux auditeurs d'entendre la façon dont la guerre est représentée à travers la langue d'un enfant-soldat.

Pour la suite, nous souhaitons perfectionner ce projet en poursuivant la lecture audio pour l'intégralité du roman de Ken Saro-Wiwa, voire en envisageant la création d'un livre audio complet pour "Allah n'est pas obligé."

Liste annexe

- **Annexe I :** contenu du fichier « content.xml » montrant une représentation brute
- **Annexe II :** contenu du fichier « content.xml » sans les éléments indésirables
- **Annexe III :** schéma récapitulant toutes les étapes de la création de l'édition enrichie du roman *Allah n'est pas obligé.*
- **Annexe IV :** schéma récapitulant toutes les étapes de la création de l'édition enrichie du roman *Sozaboy (pétit minitaire).*

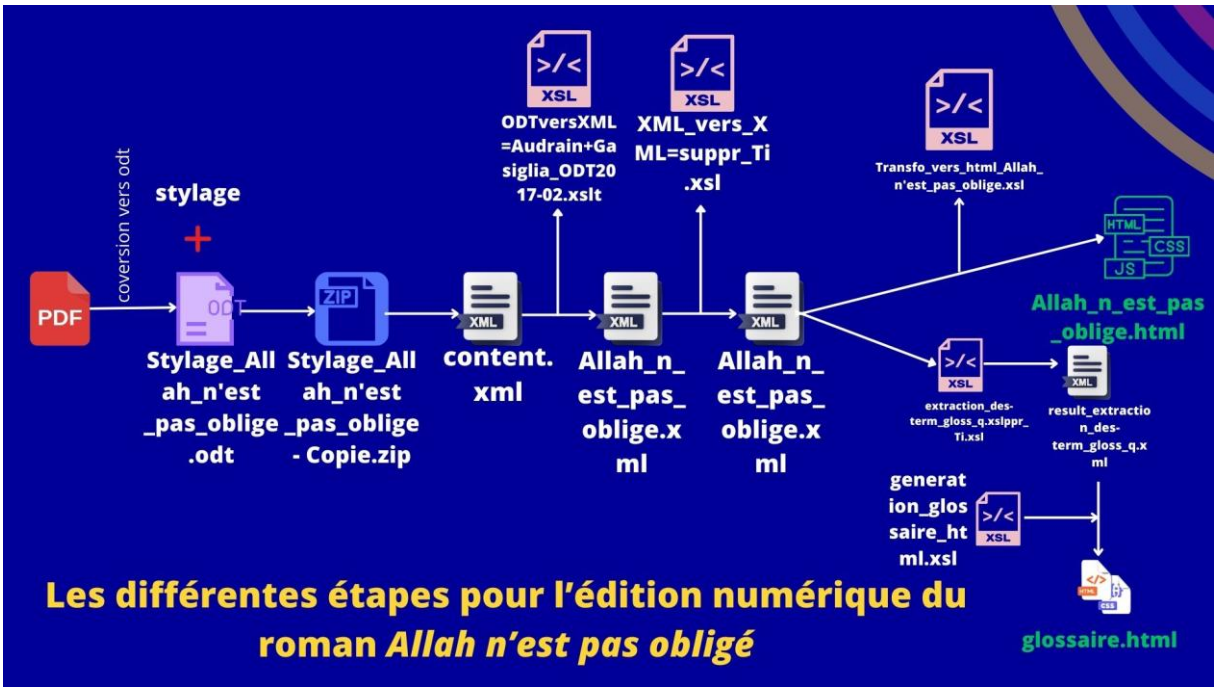
Annexe I

```
text:use-soft-page-breaks="true"><text:sequence-decls><text:sequence-decl text:display-outline-level="0"
text:name="Illustration"/><text:sequence-decl text:display-outline-level="0" text:name="Table"/><text:sequence-decl
text:display-outline-level="0" text:name="Text"/><text:sequence-decl text:display-outline-level="0"
text:name="Drawing"/></text:sequence-decls><text:p text:style-name="P5"> </text:p><text:p text:style-name="P7"> </text:p><text:p text:style-name=
"P7"> </text:p><text:p text:style-name="P7"> </text:p><text:p text:style-name="P7"> </text:p><text:p text:style-name="P7"> </text:p><text:p
text:style-name="P7"> </text:p><text:p text:style-name="P7"> </text:p><text:p text:style-name="P7"> </text:p><text:p
text:style-name="P7"> </text:p><text:p text:style-name="P7"> </text:p><text:p text:style-name="P7"> </text:p><text:p
text:style-name="P7"> </text:p><text:p text:style-name="P7"> </text:p><text:p text:style-name="P7"> </text:p><text:p
text:style-name="P7"> </text:p><text:p text:style-name="P7"> </text:p><text:p text:style-name="P10">Allah n'est pas obligé<text:span
text:style-name="T1"> </text:span>de</text:p><text:p text:style-name="Text_20_body">Ahmadou Kourouma</text:p><text:p
text:style-name="P5"> </text:p><text:p text:style-name="P2"> </text:p><text:p text:style-name="P2">Aux enfants de Djibouti : c'est à votre
demande que ce livre a été écrit.<text:span text:style-name="T2"> </text:span>Et à mon épouse, pour sa patience.</text:p><text:h text:style-name=
"P1" text:outline-level="1"><text:bookmark text:name="_I."/>I.</text:h><text:p text:style-name="P11"><text:span text:style-name="T4">Je décide
le titre définitif et complet de mon blablabla est</text:span><text:span text:style-name="T5"> </text:span><text:span
text:style-name="T15">Allah n'est pas obligé</text:span></text:p><text:p text:style-name="P14"><text:span text:style-name="T15">d'être juste
dans toutes ses choses ici-bas. </text:span><text:span text:style-name="T4">Voilà. Je commence à conter mes
```

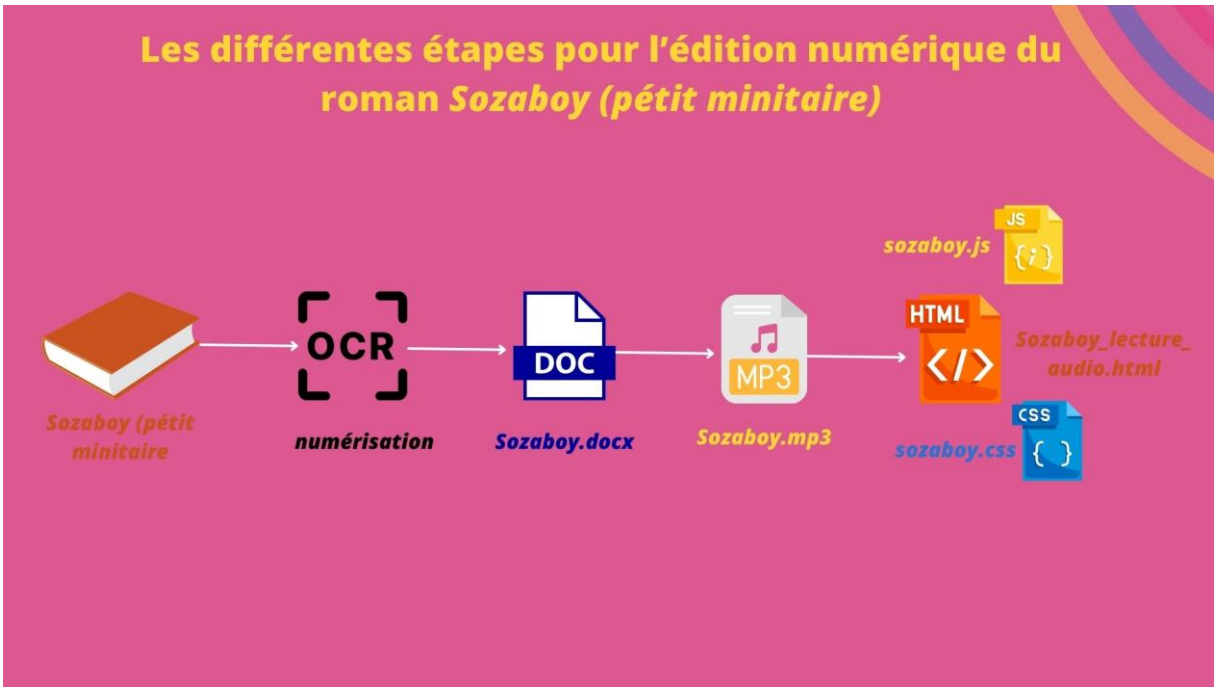
Annexe II

```
<Text_20_body> </Text_20_body>
<Text_20_body>Avant de débarquer au Liberia, j'étais un enfant sans peur ni reproche. Je dormais partout, chapardais tout et partout pour mang
<Text_20_body>les oiseaux dans la brousse. Un vrai enfant nègre noir africain broussard. Avant tout ça, j'étais un gosse dans la case avec ma
   <prbv> </prbv>
   <prbv>quelque chose comme serpent, arbre, bétail ou homme ou femme avant d'entrer dans</prbv>
   <prbv> </prbv>
   <prbv>le ventre de sa maman</prbv>. On appelle ça la vie avant la vie. J'ai vécu la vie avant la vie. Gnamokodé (bâtardise) !</Text_20_body
<Text_20_body> </Text_20_body>
<Text_20_body>La première chose qui est dans mon intérieur… En français correct, on ne dit pas dans l'intérieur, mais dans la tête. La chose
   <def> </def>
   <def>braise</def>.) Ma maman n'avait pas compté mon âge et mes mois ; elle n'en avait pas le loisir vu qu'elle souffrait tout le temps, pl
<Text_20_body>J'ai oublié de vous dire quelque chose de fondamental, de très, de formidablement important. Ma maman marchait sur les fesses.
<Text_20_body> </Text_20_body>
<Text_20_body>Donc, quand j'étais un enfant mignon, au centre de mon enfance, il y avait l'ulcère qui mangeait et pourrissait la jambe droite
<Text_20_body>Le foyer fumait ou tisonnait. (Tisonner, c'est remuer les tisons d'un feu pour l'attiser.) Autour du foyer, des canaris. (<ent>
```

Annexe III



Annexe IV



Bibliographie

Kourouma Ahmadou, *Allah n'est pas obligé*, Paris, Éditions du Seuil, 2000.

Saro-Wiwa Ken, *Sozaboy* [1985], *Sozaboy (Pétit minitaire)*, traduit de l'« anglais pourri » (Nigéria) par Samuel Millogo et Amadou Bissiri, Arles, Actes Sud, coll. « Babel », 2006.

Table des matières

Introduction 2

I. *Allah n'est pas obligé* 3

A. Récupération du texte au format ODT 3

1. Stylage du texte 4

B. Étapes de la conversion du fichier ODT en XML 5

1. Convertir le fichier ODT en un fichier XML 5

2. Méthodologie de traitement ODT vers XML 5

3. Création d'un autre fichier xml vers xml par transformation XSLT 8

C. LES ETAPES DE TRANSFORMATION VERS HTML 12

1. Élaboration d'un document HTML pour l'œuvre *Allah n'est pas obligé* avec une feuille de styles CSS. 12

D. Création d'un fichier HTML pour le glossaire. 13

1. Objectif 13

2. Création d'un code XSLT qui génère un fichier glossaire.xml 14

3. Génération d'un fichier "glossaire.html" à partir du fichier "glossaire.xml" 15

II. *Sozaboy (pétit minitaire)* 16

A. Génération d'un fichier HTML : sozaboy.html 16

1. Objectif 16

2. Étapes 16

III. Page d'accueil 20

A. Sous-menu Texte 20

B. Sous-menu Répertoire lexicographique 20

Conclusion 22

Liste annexe 23

Bibliographie 25