

---

# Cahier des charges

## PROJET : WIKIMATRIX

---

### Objet de l'étude

Réalisation d'une solution logicielle visant à extraire les tableaux de pages Wikipédia ciblées au format CSV pouvant être exploités à des fins d'analyse.

### Développeurs/Concepteurs sur le projet

- Sophie Bouvry ([sophie.bouvry1@gmail.com](mailto:sophie.bouvry1@gmail.com))
- Sullivan Dairou ([sullivan.dairou@gmail.com](mailto:sullivan.dairou@gmail.com))
- Kénan Lethimonier-Herout ([kenan.lethimonier@etudiant.univ-rennes1.fr](mailto:kenan.lethimonier@etudiant.univ-rennes1.fr))
- Margaux Boullé ([margaux.boulle@laposte.net](mailto:margaux.boulle@laposte.net))

### Contexte

Ce projet de développement logiciel est réalisé dans le cadre de l'UE PDL du Master 1 MIAGE.

### Accès aux pièces numériques

L'intégralité du code de l'application, ainsi que les diagrammes orientant son fonctionnement sont disponibles sur le repository GitHub suivant : [SulliDai/PDL\\_2018-2019\\_GR1](#)

Ce document comporte 20 pages numérotées de 1 à 20, annexes comprises.

# SOMMAIRE

## INTRODUCTION

- Objectif général à atteindre page 3
- Outils de gestion pour le projet page 3
- Langages de programmation et frameworks utilisés page 4
- Description du résultat attendu page 4

## DIAGRAMMES

- Diagramme des cas d'utilisation page 6
- Scénarios des cas d'utilisation page 7
- Diagramme de classes page 12
- Diagramme d'états-transitions page 12

## RÉALISATION

- Description de l'existant page 13
- Librairies candidates page 13
- Démarche de récupération des tableaux via HTML page 14
- Démarche de récupération des tableaux via Wikicode page 14
- Ebauche de solution page 15

## CONCLUSION

- Evaluation de la difficulté des tâches page 16
- Organisation des tâches et Gantt page 16

## ANNEXES

- Annexe 1 : Charte utilisateur page 17
- Annexe 2 : Prise en charge des tableaux page 18

# INTRODUCTION

## → Objectif général à atteindre

Wikipédia est une encyclopédie numérique collaborative universelle. Une communauté importante consulte et alimente Wikipédia régulièrement, les données présentes dans le site sont donc difficilement exploitables du fait de leur hétérogénéité. Ainsi, chaque contributeur écrit lui-même le wikicode de ses pages, ce qui peut entraîner de grandes divergences entre les pages. De plus, il n'est pas facile actuellement d'extraire des tableaux, les solutions existantes étant rudimentaires (Ex : copier/coller) et limitées.

Wikimatrix permettra l'exportation d'un maximum de données tabulaires de Wikipédia vers des fichiers de format CSV. La force de ce projet est de solutionner l'hétérogénéité des différents tableaux. L'extraction permettra une meilleure réutilisation des données, notamment dans les domaines de la statistique et de l'analyse. Wikipédia étant une plateforme collaborative, le projet sera Open Source, afin de rester dans un esprit de partage.

## → Outils de gestion pour le projet

- **Trello** : Cet outil permet la gestion en ligne de projets sous la forme d'un tableau. Le tableau est découpé en catégories (en cours, à faire, terminé...) listant des tâches représentées sous forme de cartes. Une carte est assignable à des utilisateurs et est mobile d'une catégorie à une autre. Il est possible de définir des priorités aux tâches et des dates de fin. Pour ce projet nous avons utilisé la version gratuite.
- **Slack** : Cette plateforme permet d'échanger sur l'avancement du projet et centraliser l'ensemble des applications que nous utilisons (Google Drive, GitHub ...)
- **Google Drive** : Il nous permet de centraliser, d'échanger et de produire collaborativement des documents liés au projet. Nous l'avons notamment choisi pour produire ensemble le présent document.
- **GitHub** : C'est un service web qui permet de versionner notre projet. Le logiciel Git gère les avancées du code grâce à plusieurs commandes telles que :
  - *Pull* : Rattrape le code du repository distant vers le repository local.
  - *Commit* : Valide le code modifié sur le repository local.
  - *Push* : Envoyer les commit locaux vers le repository distant.
  - *Fetch* : Récupère l'organisation par branches du repository distant.

Le grand avantage d'une solution Git est de pouvoir réaliser des branches, cela permet d'éviter de détruire le code des autres développeurs.

Après délibération, nous avons décidé de segmenter le projet en plusieurs versions par l'intermédiaire de tags. À chaque fois qu'une version du projet est fonctionnelle, cela sera indiqué par un tag. Plusieurs arguments nous ont amenés à ce choix :

- Le projet contient peu de classes (seulement 8).

- L'ensemble des classes seront modifiées par tous les développeurs en même temps. Le développement d'une classe dépend des autres, il est donc difficile de travailler avec des branches. Cependant cette organisation pourra être modifiée selon les difficultés du projet.

De plus si nous constatons qu'une portion du code acceptée est erronée nous pouvons revenir vers une version plus stable qui aura été taguée.

Pour ce qui est de l'organisation, le code possède un dossier src qui contiendra le programme et le code de test afin de vérifier la bonne application des méthodes.

- **Maven** : a également retenu notre attention, car il est compétent sur les points de gestion et d'automatisation de la production des projets logiciels. Cet atout permet de créer un fichier pom.xml qui chargera et intégrera les différentes librairies dans le projet. Cela permet de gérer les extensions et les dépendances de manière automatisée.

- **Travis CI** : Ce logiciel d'intégration continue est intégré dans GitHub. Il a pour objectif de réaliser automatiquement les tests de fonctions après chaque push. Ce module aura comme impact de détecter facilement les rétrogradations involontaires de l'application. Par exemple, lorsque l'on va modifier notre code afin de traiter des tableaux plus compliqués, il va être très utile. Il nous enverra des mails en cas de tests qui ne fonctionnent plus. De plus, sur notre github, un label sera présent pour indiquer que notre code a été analysé par cette application, c'est aussi un gage de confiance pour les futurs utilisateurs. Cette application est caractérisée par son fichier .travis.yml qui centralise la méthode de traitement par le logiciel.

### → Langage de programmation et frameworks utilisés

- **Java** : Java est un langage de programmation orienté objet. Cet aspect du projet nous est imposé par le client. Ce choix vient du fait que dans notre formation le développement sous java est le plus répandu. De plus nous avons plus de repères sur l'environnement de Java, notamment avec Junit, JavaFX... De plus, Java a la particularité de pouvoir s'exécuter sous une multitude de plateformes.

- **Junit** : C'est un framework de tests unitaires pour le langage de programmation Java. Ce framework permet de réaliser des tests de conformité des objets et méthodes créés. Notamment, dans notre projet, vérifier que les CSV sont conformes avec le cahier des charges.

### → Description du résultat attendu

A terme, l'utilisateur pourra demander l'extraction, par l'intermédiaire d'une ligne de commande, d'une ou plusieurs URLs (instanciation des URLs automatique via un fichier txt). Pour interagir avec la machine, l'utilisateur aura à sa disposition une liste de commandes. Il existe une charte utilisateur (voir annexe 1) auquel il est possible de se référer. Il pourra demander à récupérer les tableaux à partir du format HTML, du Wikicode ou des deux en même temps.

Deux cas se présentent alors :

→ **L'URL ne peut être traitée, pour plusieurs raisons :**

- Le domaine Wikipédia ne peut être pris en compte, nous ne considérons (pour une première version) que les adresses françaises et anglaises Wikipédia, en écartant par exemple les versions régionales.

- L'URL ne contient pas de tableaux ou elle ne contient pas de types de tableaux pris en charge par Wikimatrix (voir annexe 2). Dans le cas où une URL présente plusieurs tableaux, l'application traitera chaque tableau indépendamment.

L'utilisateur sera informé de la non prise en compte de sa demande par un message.

→ **L'URL peut être traitée :**

- L'utilisateur obtiendra en sortie un fichier CSV pour chaque tableau présent dans l'URL. Si il sélectionne les deux formats, il obtiendra un CSV pour chaque tableau extrait. Des messages pourront être affichés à l'utilisateur au cours de l'extraction ( Ex : Cellules fusionnées supprimées ...).

**L'application se devra d'être :**

- *Fonctionnelle : Traiter un maximum de données tabulaires*
- *Facile d'utilisation : Commandes explicites*
- *Fiable : La fiabilité sera notamment testée avec notre comparateur*
- *Performante : Rapidité d'exécution*
- *Maintenable : Modularité du code, notamment pour une future extension.*
- *Portable : Adaptable à différents OS (Windows, Linux, MacOS...)*
- *Open source : Libre d'accès, documentée.*

Les tableaux Wikipédia devront être extraits à partir de deux formats :

- **HTML** : Chaque page web est constituée de ce langage à balises.
- **Wikicode** : Présent dans chaque page Wikipédia.

L'intérêt pour le client est qu'il peut générer le CSV le plus adapté à son étude. Les deux langages étant différents, ils ne contiennent pas forcément les mêmes informations.

**Dans l'objectif de répondre au mieux aux besoins de l'utilisateur, plusieurs options lui seront proposées :**

- *Choix du nom du fichier*
- *Choix de l'importation : une liste d'URL ou une simple URL*
- *A partir de quel format l'extraction est-elle effectuée : html, Wikicode ou les deux ?*
- *Le choix de l'emplacement du fichier*
- *La délimitation du CSV*

Chaque option possédera une valeur par défaut tel que la délimitation du CSV qui sera par défaut « , ».

## DIAGRAMMES

### → Diagramme de cas d'utilisation

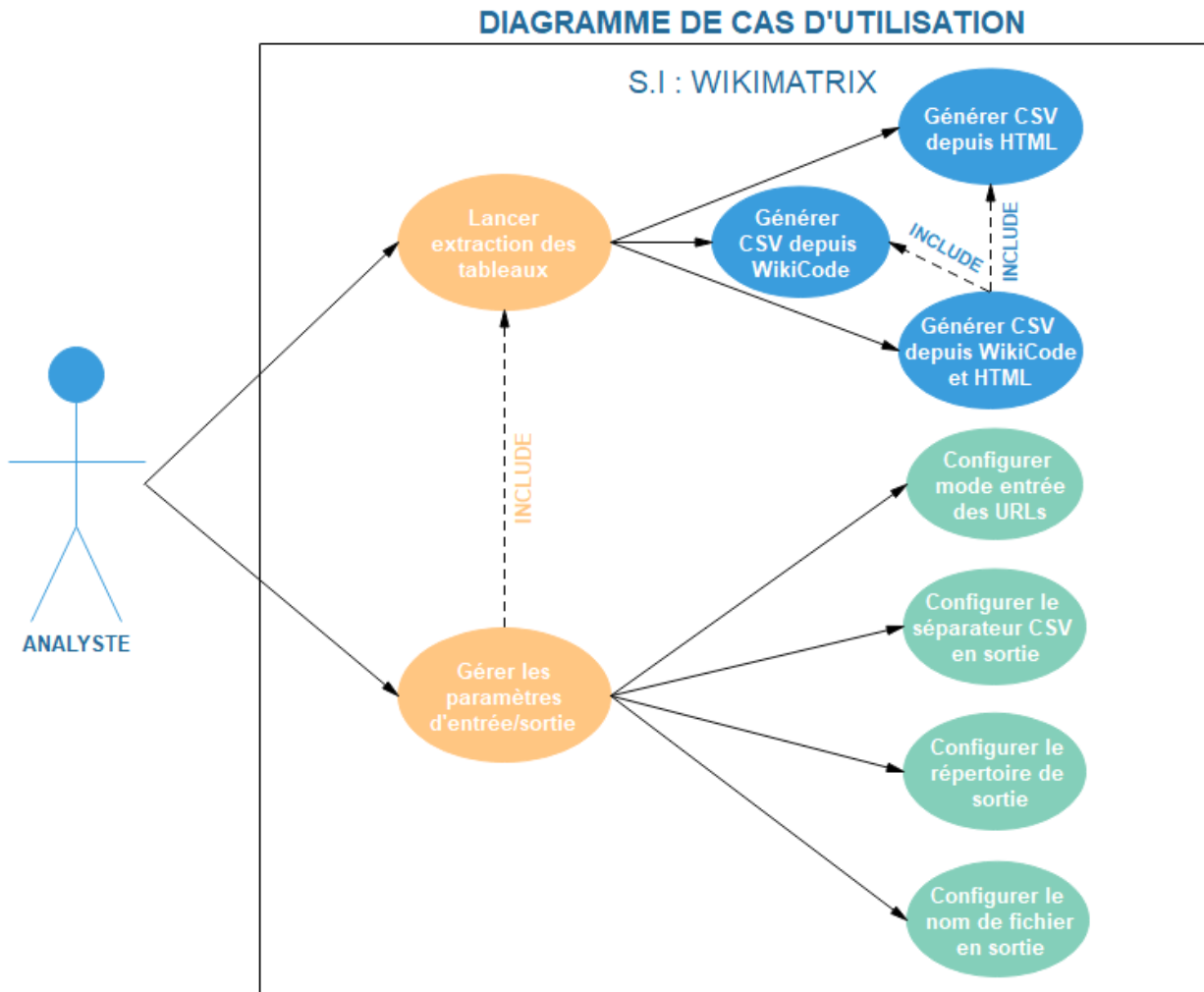


Illustration 1: Diagramme de cas d'utilisation

Le système offre la possibilité de choisir le langage à partir duquel les données seront extraites des tableaux Wikipédia. Chaque paramètre d'entrée/sortie dispose d'une valeur par défaut (se référer à la charte utilisateur). Ces dernières peuvent être modifiées à l'aide d'options à ajouter à la ligne de commande qui initie l'extraction des tableaux.

## 1 : Configurer le répertoire de sortie

**Acteur(s) :** Le client.

**Description :** L'application propose un chemin de sauvegarde pour le fichier par défaut. L'utilisateur a néanmoins la possibilité de changer ce chemin en ajoutant une commande lors de l'importation de l'URL.

**Précondition :** La commande d'importation du fichier ou d'une URL doit être valide.

**Démarrage :** Le client lance l'application.

### Description

#### Le scénario nominal

- 1) Le système attend une instruction de la part de l'utilisateur.
- 2) L'utilisateur tape une ligne de commande commençant par l'import d'une URL ou d'un fichier .txt contenant une multitude d'URL. Ensuite il saisit la commande permettant de configurer le chemin de sauvegarde du fichier.
- 3) Vérification de la validité de la ligne de commande.
- 4) Le système vérifie l'accessibilité du chemin indiqué.
- 5) Le système traite.
- 6) Le système enregistre le fichier CSV créé à l'adresse fournie par l'utilisateur suivi du nom du langage utilisé par l'extraction des données.

#### Les scénarios d'exception

- 3)a Une des commandes présentes dans la ligne de commande n'est pas valide. Le système affiche un message d'erreur et attend une nouvelle instruction.
- 4)a Si le chemin n'est pas accessible car l'utilisateur n'a pas les droits pour cet emplacement ou car le chemin n'existe pas, le système affiche un message d'erreur à l'utilisateur et attend une nouvelle instruction.

#### Les scénarios alternatifs

- 2)a L'utilisateur ne renseigne pas la commande permettant la configuration du chemin de sauvegarde du fichier. Dans ce cas le chemin de sauvegarde sera le chemin par défaut de l'application.

**Fin :** Scénario d'exception 3)a et 4)a

**Post condition :** Le fichier CSV généré est sauvegardé à l'endroit indiqué par l'utilisateur.

## 2 : Générer CSV depuis HTML et wikicode

**Acteur(s)** : Le client.

**Description** : Le client à la possibilité de générer deux fichiers CSV, l'un depuis le code HTML d'une page Wikipédia, l'autre puis le HTML depuis la même page.

**Précondition** : /

**Démarrage** : Le client lance l'application.

### Description

#### Le scénario nominal

- 1) Le système attend une action de la part du client.
- 2) Le client renseigne une ligne de commande qui importe un fichier .txt contenant une multitude d'URL ou une simple URL. La commande doit également renseigner le type d'importation des données grâce aux commandes spécifiques. Le client peut se référer à la charte utilisateur.
- 3) Le système vérifie la validité de la ligne de commande.  
Les commandes doivent exister, l'URL doit être une URL Wikipédia.
- 4) Le système récupère le code HTML de la page.
- 5) Le système vérifie la présence de tableaux dans la page.
- 6) Pour chaque tableau présent, le système vérifie leur éligibilité au traitement. Il doit exclure les tableaux trop complexes du processus.
- 7) Le système traite les tableaux un par un afin d'en extraire les données.
- 8) Le système crée un fichier CSV, par tableau, et le génère dans un répertoire par défaut. Le nom du fichier est par défaut le nom du tableau suivi du langage utilisé lors de l'extraction des données.
- 9) Le système recommence l'étape 4 avec le Wikicode.

#### Les scénarios d'exception

- 3)a L'URL n'est pas valide ou la commande n'existe pas. Le système informe l'utilisateur avec un message d'erreur et attend de nouvelles instructions.

#### Les scénarios alternatifs

- 2)a Aucun tableau n'est présent dans la page renseignée à travers l'URL. Le système informe l'utilisateur de l'absence de tableaux dans le cas d'une URL seule avec un message d'erreur et attend une instruction. Dans le cas d'un fichier avec une multitude d'URL le système continue son traitement avec l'URL suivante et affichera un message d'information à la fin des traitements.
- 6)a Un tableau trop complexe à la conversion en CSV ne doit pas être traité, dans ce cas le système garde en mémoire un message d'information qu'il délivrera à l'utilisateur à la fin de tout traitement.
- 7)a Pour des données complexes comme des fusions de cellules, la ligne est supprimée. Le système garde en mémoire un message d'information à destination de l'utilisateur, il lui restituera à la fin de tout traitement.

**Fin** : Scénario d'exception à l'étape 3)a et scénario alternatif 2)a

Post condition : 1 fichier CSV généré à partir du HTML, 1 fichier généré à partir du wikicode pour chaque tableau



### 3 : Configurer mode d'entrée URL

**Acteur(s)** : Le client.

**Description** : Le client a la possibilité de configurer le mode d'entrée des URL. Les URL wikipédia peuvent être importées grâce à un fichier ou entrées une à une grâce aux lignes de commandes.

**Précondition** : /

**Démarrage** : Le client lance l'application.

#### Description

##### Le scénario nominal

- 1) Le système attend une directive de la part de l'utilisateur.
- 2) L'utilisateur entre la commande afin de renseigner une adresse URL ou un fichier contenant une multitude d'adresses.  
La commande doit commencer par l'import d'un fichier ou d'une URL.
- 3) Le système interprète la commande et vérifie les adresses URL.

##### Les scénarios d'exception

- 2)a La commande renseignée n'existe pas, dans ce cas un message d'aide s'affiche à l'écran.
- 3)a Si une ou plusieurs des adresses n'est pas une adresse wikipédia, le système informe l'utilisateur que les adresses sont erronées avec un message d'erreur, il attend une nouvelle instruction de la part de l'utilisateur.

##### Les scénarios alternatifs

**Fin** : Scénario d'exception au 2)a et 3)a

**Post condition** : Le système prend en compte l'import soit d'un fichier soit d'une URL seule.

## 4 : Configurer le nom du fichier CSV en sortie

**Acteur(s) :** Le client.

**Description :** L'application propose un nom par défaut pour le ou les fichiers en sortie. L'utilisateur a néanmoins la possibilité de changer ce nom en ajoutant une commande lors de l'importation de l'URL.

**Précondition :** L'utilisateur doit indiquer un nom de fichier qui n'est pas déjà utilisé dans le répertoire courant.

**Démarrage :** Le client lance l'application.

### Description

#### Le scénario nominal

- 1) Le système attend une instruction de la part de l'utilisateur.
- 2) L'utilisateur tape une ligne de commande commençant par l'import d'une URL ou d'un fichier .txt contenant une ou plusieurs URLs suivi de la commande permettant de configurer un nom de fichier personnalisé pour sauvegarder le fichier CSV généré en sortie.
- 3) Le système vérifie l'intégrité de la ligne de commande.
- 4) Le système vérifie qu'aucun fichier du même nom existe déjà dans le répertoire courant.
- 5) Le système traite l'extraction des tableaux à l'URL indiquée.
- 6) Le système enregistre le fichier CSV sous le nom fourni par l'utilisateur.

#### Les scénarios d'exception

- 3)a Une des commandes présentes dans l'instruction de l'utilisateur n'est pas valide. Le système affiche un message d'erreur et attend une nouvelle instruction, toujours en ligne de commande.
- 4)a Si le nom choisi par l'utilisateur est déjà utilisé par un autre fichier dans le répertoire courant, le système affiche un message à l'utilisateur et incrémente un nom valide.

#### Les scénarios alternatifs

- 2)a L'utilisateur ne renseigne pas la commande de configuration du nom du fichier en sortie. Dans ce cas, le nom du fichier suivra la configuration par défaut de l'application, et portera pour nom le nom du tableau concerné.

**Fin :**

**Post condition :** Le fichier CSV généré est sauvegardé sous le nom indiqué par l'utilisateur.

## 5 : Configurer le séparateur CSV en sortie

**Acteur(s) :** Le client.

**Description :** Le client a la possibilité de configurer le séparateur qui sera utilisé par le système pour la génération du fichier CSV en sortie, en ajoutant une commande lors de l'importation de l'URL.

**Précondition :** L'utilisateur devra indiquer un séparateur inclus dans la liste des séparateurs autorisés par le système.

**Démarrage :** Le client lance l'application.

### Description

#### Le scénario nominal

- 1) Le système attend une instruction de la part de l'utilisateur.
- 2) L'utilisateur tape une ligne de commande commençant par l'import d'une URL ou d'un fichier .txt contenant une ou plusieurs URLs suivies de la commande permettant de configurer le séparateur CSV pour le fichier généré par le système en sortie.
- 3) Le système vérifie l'intégrité de la ligne de commande.
- 4) Le système vérifie que le séparateur choisi par l'utilisateur est bien un séparateur autorisé par le système.
- 5) Le système traite l'extraction des tableaux à l'URL indiquée.
- 6) Le système génère et enregistre le fichier CSV à l'aide du séparateur indiqué par l'utilisateur.

#### Les scénarios d'exception

3)a Une des commandes présentes dans l'instruction de l'utilisateur n'est pas valide. Le système affiche un message d'erreur et attend une nouvelle instruction, toujours en ligne de commande.

4)a Si le séparateur choisi par l'utilisateur est interdit par le système (dans un souci de cohérence du fichier CSV généré), le système affiche un message à l'utilisateur et le système choisit un séparateur valide.

Si dans les données, il y a le séparateur, l'utilisateur sera informé par un message d'erreur pour relancer la commande concernée.

4)b Si c'est le système qui choisit le séparateur et qu'il y a une donnée qui l'utilise le système en choisira un autre.

#### Les scénarios alternatifs

2)a L'utilisateur ne renseigne pas la commande de configuration du séparateur pour la génération du CSV en sortie. Dans ce cas, le séparateur utilisé par le système prendra sa valeur par défaut, c'est à dire « ; ».

**Fin :**

**Post condition :** Les différents éléments du fichier CSV en sortie sont bien séparés à l'aide du séparateur indiqué par l'utilisateur.

## → Diagramme de classes

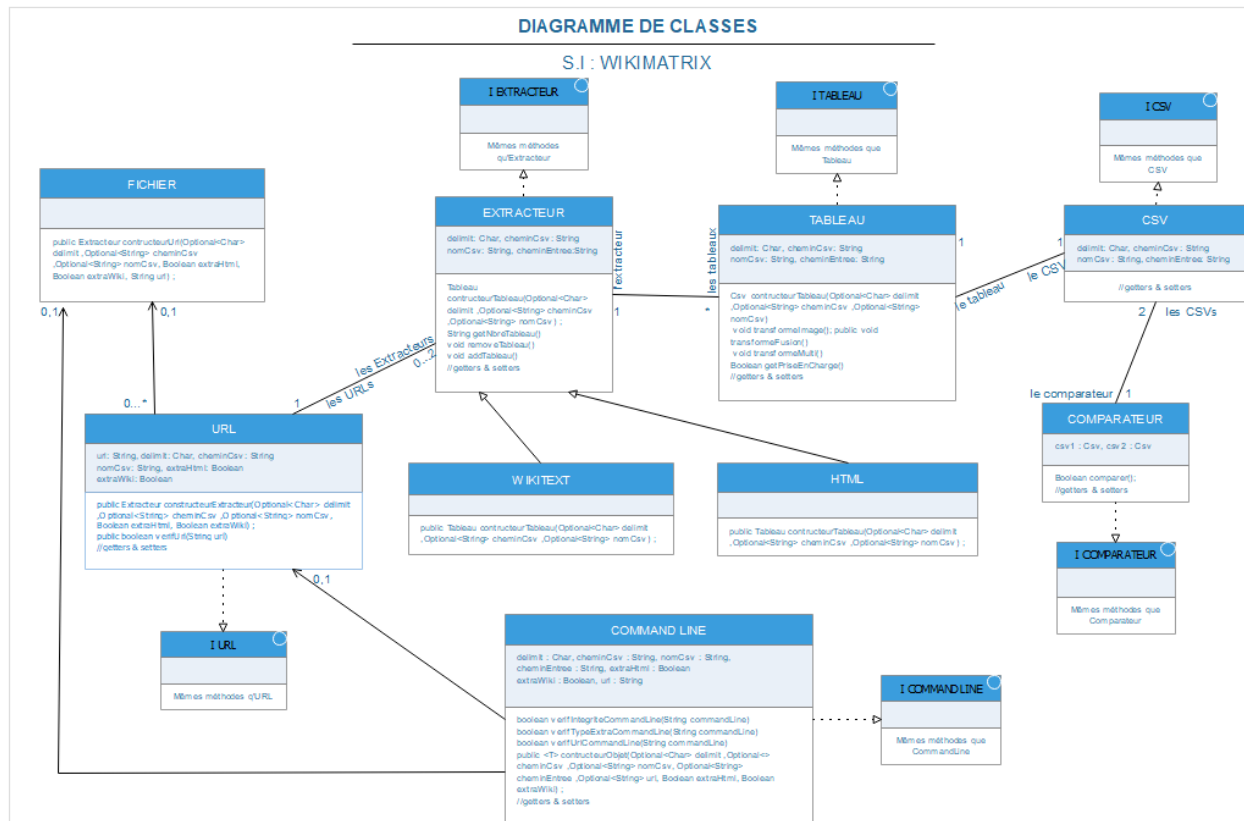


Illustration 2: Diagramme de classes

## → Diagramme d'états-transitions

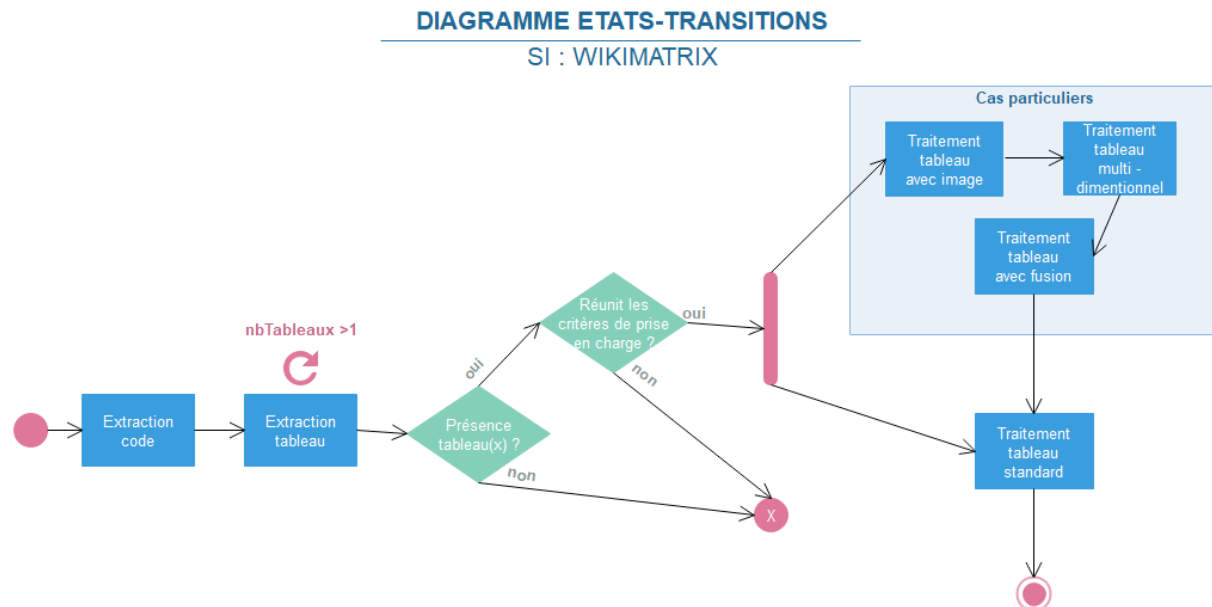


Illustration 3: Diagramme d'états-transitions

## RÉALISATION

### → Description de l'existant

Plusieurs projets ayant pour but d'exploiter les données existent. Un des plus importants est Wikidata, son objectif est de structurer les informations présentes dans Wikipédia. Une base de données regroupe des statistiques et des informations de différentes versions linguistiques, facilement manipulables par des machines, des utilisateurs. Ce projet est collaboratif, et plus de 19.000 utilisateurs sont actifs pour mettre à jour et augmenter la base de données. De nombreux projets se sont développés à partir de celui-ci, notamment pour extraire les données du projet Wikidata.

DBpedia, qui est un projet communautaire, qui permet quant à lui d'extraire des informations structurées de Wikipédia pour les rendre disponibles sur internet au format RDF (sous forme de graphes). Plusieurs solutions sont proposées pour avoir accès aux données RDF :

- Directement par URL
- Utilisation d'agents Web
- Points d'accès SPARQL

Des solutions existent mais aucune ne s'intéresse au traitement particulier que nécessite les tableaux et à leur besoin d'uniformisation. Notre projet s'inscrivant dans un contexte de partage, il devra avoir une documentation complète pour favoriser sa réutilisabilité. Pour cela nous allons, à la fin, mettre à disposition la javadoc de ce dernier.

### → Librairies candidates

#### Parseurs HTML :

Librairie	Maintenue ?	Documentée ?	Open source ?	Connue?
<b>Jsoup</b>	<b>Dernière version avril 2018</b>	<b>oui</b>	<b>oui</b>	<b>oui</b>
HTMLCleaner	Dernière version avril 2018	oui	oui	non
Lagarto	Dernière version octobre 2018	oui	oui	non
HTML Parser 2.0	non plus depuis 2006	oui	oui	non

### Parseurs Wikicode :

Librairie	Maintenue ?	Documentée ?	Sortie	Open source ?
<b>Sweble</b>	<b>oui</b>	<b>oui</b>	<b>Abstract syntax tree</b>	<b>oui</b>
mediawiki.japi	oui	oui	XML / Json	oui
jwiki	oui	pas super bien	String	oui

#### → Démarche de récupération des tableaux via l'HTML

Nous avons choisi d'utiliser la librairie Jsoup car c'est une des librairies les plus connues pour parser le HTML. Elle est régulièrement maintenue et ses fonctionnalités correspondent parfaitement à besoins. Avec la fonction testJsoup, nous avons réussi à extraire les tableaux du code HTML.

```
public static void testJsoup() throws IOException {
    Document doc =
```

```
Jsoup.connect("https://en.wikipedia.org/wiki/Comparison_of_Canon_EOS_digital_cameras").get()
;
    Element table = doc.select("table.wikitable").get(0);
    Elements rows = table.getElementsByTag("tr");
    for (int i = 1; i < rows.size(); i++) { // first row is the col names so skip it.
        Element row = rows.get(i);
        Elements cols = row.select("td");
        String d = cols.toString();
    }
}
```

#### → Démarche de récupération des tableaux via le Wikicode

Tout d'abord, nous nous étions concentrés sur une première solution qui n'a pas été retenue. Cette dernière consistait à récupérer le wikicode en passant par l'URL de l'utilisateur et la transformer en URL éditeur.

Exemple :

**URL d'origine :** [https://en.wikipedia.org/wiki/Comparison\\_of\\_Canon\\_EOS\\_digital\\_cameras](https://en.wikipedia.org/wiki/Comparison_of_Canon_EOS_digital_cameras)

**URL adaptée :**

[https://en.wikipedia.org/Comparison\\_of\\_Canon\\_EOS\\_digital\\_cameras?action=edit](https://en.wikipedia.org/Comparison_of_Canon_EOS_digital_cameras?action=edit)

La conversion de l'URL permet de capturer le wikicode dans la balise "textarea" du code HTML.

Sweble permet de récupérer le wikicode sous forme d'arbre, c'est la solution que nous avons privilégié. Grâce à cela nous pouvons parcourir chaque noeud de l'arbre et chercher les tableaux "wikitable". Les fonctions ci-dessous sont une première ébauche de la solution d'extraction des tableaux avec le Wikicode.

```
private static void parcourirNode(WtNode fils) {
    for (Iterator<WtNode> I = fils.iterator(); I.hasNext();) {
        fils = I.next();
        if (fils.getNodeName().toString().equals("WtTable")) {
            System.out.println(fils.getNodeName().toString());
            String wikitable = fils.toString();
            if(wikitable.indexOf("wikitable")!=-1){
                //Inserion dans une liste de tableau
                parcourirNode(fils);
            }
        }
    }
}
```

#### → Ebauche de solution

Une première ébauche pour vérifier l'intégrité des lignes de commandes saisies par l'utilisateur à été développée.

Nous avons également développé une première solution pour la création de fichiers CSV.

*Nous vous invitons à consulter notre repository [GitHub](#) pour en prendre connaissance.*

## CONCLUSION

### → Evaluation de la difficulté des tâches

La première approche nous a montré une difficulté sur le choix et la mise en place des librairies. Il en existe une multitude qui peuvent être difficiles à exploiter. C'est pourquoi le choix des librairies fût important, car elles auront un impact sur la suite du développement. Après avoir étudié les tableaux, nous avons relevé plusieurs points qui peuvent poser problème. Pour notre implémentation, nous allons commencer par traiter les tableaux standards. Nous traiterons les cas plus complexes dans des versions ultérieures.

La principale difficulté technique est l'hétérogénéité de la structure des tableaux. Reconnaître les tableaux qui ne valident pas les critères de prise en charge (voir Annexe2) est un enjeu majeur. L'autre difficulté sera de traiter les tableaux "non-standard" répondant aux critères de validité.

### → Organisation des tâches et planning prévisionnel de Gantt

A travers un diagramme de Gantt nous avons pu mettre en avant les principales tâches à accomplir pour ce projet. Nous avons défini des temps limités pour réaliser ces dernières. Chacune de ces tâches à un ordre de priorité pour leur réalisation. Ce planning est amené à évoluer tout au long du projet, en fonction des difficultés que nous rencontrerons.

*Les diagrammes de Gantt et de PERT sont disponibles sur le repository GitHub.*



## ANNEXES

### → Annexe 1 : Charte utilisateur

Cette charte regroupe l'ensemble des commandes à saisir dans le terminal pour que l'utilisateur puisse interagir avec l'application.

Pour réaliser une commande, il faut obligatoirement satisfaire la condition :

→ Renseigner une ou plusieurs URLs.

*Les commandes à renseigner dans le terminal sont en gras.*

→ Importer un fichier .txt où sont rangées l'ensemble des URLs :

**-import[C://AdresseFichier] :**

Lorsqu'on utilise la commande **-import[C://AdresseFichier]** cela extrait la liste des URLs d'un fichier. Afin de normaliser le fichier d'entrée, il a été décidé dans la première version de notre application de se limiter au format "txt".

À l'intérieur de ce fichier, les données seront délimitées par le caractère " ; ". Ainsi chaque URL sera séparée par ce caractère.

**-url[adresse] :**

→ Choisir extraction html et/ou Wikicode

**-html** : Extraction des données par le code HTML

**-wikicode** : Extraction des données par le code Wikicode

→ Choisir le délimiteur dans le CSV final

**-delimit[,]** :

→ Enregistrer le/les CSV

**-save[c/]** :

→ Enregistrer le/les CSV sous quel nom

**-name[fichier.csv]** :

*Attention, les commandes **import** et **url** ne peuvent pas être réalisées en même temps.*

*Les commandes **-html** et **-wikicode** peuvent être cumulées pour avoir les deux extractions simultanément sur les mêmes tableaux.*

### Exemples de commandes acceptées par l'application :

>>WikiMatrix -import[C:\Users\Sophie\Documents\Dossier cours\liens.txt] -html

Cette commande permet la création de fichiers CSV à partir des URL présentent de "liens.txt". L'extraction sera fera à partir du code HTML. Tous les autres paramètres seront définis par leur valeur par défaut.

>>WikiMatrix -url[https://fr.wikipedia.org/wiki/Rennes] -html -wikicode -delimit[;] -save[C:\Users\Sophie\Documents\]

Le programme extrait l'URL et analyse la page avec les deux algorithmes. A l'aide de cette commande, l'utilisateur paramètre le délimiteur et l'emplacement de sauvegarde.

### → Annexe 2 : Prise en charge des tableaux

#### 1) Tableaux simples

- **Tableau standard**

Ce type de tableau sera pris en compte sans difficulté particulière.

Titre			
	Titre col. A	Titre col. B	Titre col. C
Titre ligne 1	Donnée 1A	Donnée 1B	Donnée 1C
Titre ligne 2	Donnée 2A	Donnée 2B	Donnée 2C
Titre ligne 3	Donnée 3A	Donnée 3B	Donnée 3C

*Illustration 4: Exemple de tableau "standard"*

Un tableau standard est un tableau ne rencontrant pas les spécificités ci-dessous. Dans le cas où un tableau présente une légende, on ne récupérera pas la donnée.

- **Tableau avec des images**

On conservera l'URL des images et quand cela est possible on l'intégrera au CSV. Dans le cas d'une colonne regroupant un texte et une image, on créera une nouvelle colonne nommée « Image » (ou avec un nom aléatoire).

Fédération	Tour 1	Tour 2
 Russie	9	13
 Espagne -  Portugal	7	7
 Belgique -  Pays-Bas	4	2
 Angleterre	2	-
Total	22	22

*Illustration 5: Exemple de tableau "avec images"*

Dans cet exemple, nous conserverons la colonne « Fédération » et nous ajouterons une nouvelle colonne avec les images des drapeaux.

- **Tableau triable**

Les tableaux de ce type seront traités comme des tableaux standards. L'extraction de données ne prendra pas en compte la possibilité de trier le tableau et affichera les valeurs de ce dernier telles qu'elles sont présentes dans le code source.

Un exemple de tableau triable est présent l'illustration 6.

- **Tableau avec des cellules fusionnées**

Deux cas se présentent :

→ Tableau hybride simple

Ces tableaux ne seront pas pris en compte. On supprime les lignes des cellules fusionnées et on affiche le reste du tableau. Un message informera l'utilisateur qu'une ou plusieurs lignes du tableau a/ont été supprimée(s).

Benelux			
Pays	Rang	Population	Date du relevé
Pays-Bas	1	16 500 000	2003
Belgique	2	10 millions	2007
Luxembourg	3	0,5 million	
Benelux	Total : 27 M		

*Illustration 6: Exemple de tableau "hybride simple" (particularité d'être triable)*

Dans ce cas, la ligne de cumule 'Benelux' sera supprimée mais le reste du tableau sera traité.

→ Tableau hybride complexe

Présentant un nombre trop important de cellules fusionnées, ils ne seront pas traités, l'utilisateur sera prévenu par un message.

Fédération	Tour 1	Tour 2
 Russie	9	13
 Espagne -  Portugal	7	7
 Belgique -  Pays-Bas	4	2
 Angleterre	2	-
Total	22	22

*Illustration 7: Exemple de tableau "hybride complexe"*

## 2) Tableaux complexes

- **Tableau multidimensionnel**

Ces tableaux seront pris en charge dans une seconde version de l'application.



Saison	Club	Championnat			Coupe nationale		Coupe de la Ligue		Supercoupe		Compétition(s) continentale(s)			Maroc		Total	
		Division	M	B	M	B	M	B	M	B	C	M	B	M	B	M	B
2007-2008	 Wydad de Casablanca	1	2	1	-	-	-	-	-	-	-	-	-	-	-	2	1
2008-2009	 Wydad de Casablanca	1	14	1	-	-	-	-	-	-	-	-	-	-	-	14	1
2009-2010	 RC Lens	1	-	-	-	-	1	0	-	-	-	-	-	2	0	3	0
2009-2010	 FC Nantes (prêt)	2	8	0	-	-	-	-	-	-	-	-	-	-	-	8	0
2010-2011	 Qadsia SC	1	18	0	-	-	-	-	-	-	C1	7	0	-	-	25	0
2011-2012	 Vitória Guimarães	1	25	0	1	0	3	0	1	0	C3	4	0	2	0	36	0
2012-2013	 Vitória Guimarães	1	24	1	5	0	2	0	-	-	-	-	-	12	1	43	2
2013-2014	 Levante UD	1	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0
Total sur la carrière			91	3	6	0	6	0	1	0	-	11	0	16	1	131	4

Illustration 8: Exemple de tableau "multidimensionnel"

- **Tableau sans identifiant wikitable**

Dans le cas où il manque l'identifiant "wikitable" présent dans chaque tableau, on ne traitera pas le tableau. L'utilisateur sera informé de l'impossibilité d'extraire des données.

Exemple :

[https://fr.wikipedia.org/wiki/David\\_Kaczowka](https://fr.wikipedia.org/wiki/David_Kaczowka)

Le wikicode du tableau "statistiques" ne contient pas l'identifiant "wikitable"  
Nous traiterons ce cas particulier dans un second temps.