Cahier des charges

1. Présentation du projet
    1.1. Contexte
    1.2. Objectifs
    1.3. Ressources et utilisation de l'existant
    1.4. Critères d'acceptabilité du projet
2. Expression des besoins
    2.1. Besoins fonctionnels
    2.2. Besoins non fonctionnels
3. Contraintes
    3.1. Délais
    3.2. Autres contraintes
4. Risques
    4.1. Evaluation
    4.2. Traitements possibles
5. Déroulement du projet
    5.1. Planification
    5.2. Documentation
        5.2.1. UML
        5.2.2. Scénario

1. Présentation du projet

1.1. Contexte

Ce projet est réalisé dans le cadre du module PDL de Master 1 MIAGE à l'université de Rennes 1.
Le groupe de projet est constitué de Vadim Kupratsevitch, Adrien Wacquet, Gaëtan Scrimali, Guillaume Mande, Thibaut Legroux.
Wikipedia est une encyclopédie libre en ligne. La base de données est donc conséquente.
L'exploitation de cette base de données est de faite complexe pour des logiciels de traitements statistiques, de visualisation ou tous autres outils d'exploitation de tableaux.
Les tableaux Wikipedia sont écrits dans une syntaxe propriétaire appelée Wikitext, elle est de faite difficile à analyser et non nécessairement conçue pour la spécification de tableaux.

1.2. Objectifs

L'objectif de ce projet de PDL est d'extraire des tableaux au format CSV à partir de pages WIKIPEDIA. Une exploitation de ces tableaux sera ensuite faite afin d'en extraire les données.

1.3. Ressources et utilisation de l'existant

A l'heure actuelle les différentes pages de Wikipedia représentent la seule ressource à notre disposition.
Nous utiliserons par la même occasion le gestionnaire de version Git afin de partager notre code et organiser le déroulement de notre projet.
Afin de réaliser les différents diagrammes UML nous utiliserons ASTAH. Par la suite nous utiliserons plusieurs IDE tel que Eclipse ou IntelliJ. Notre programme sera bien entendu codé en Java.
De plus nous utiliserons plusieurs librairies existantes telles que Jsoup pour le parsing HTML et Sweble pour le parsing du Wikitext.
Nous avons choisi Jsoup pour le parsing HTML tout d'abord car cette librairie existe depuis 2010 et continue de recevoir des mises à jours fréquentes qui enrichissent les façons de sélectionner des éléments sur la page. En utilisant cette librairie, il est possible de manipuler les données d'une page en utilisant des sélecteurs CSS, des méthodes semblables à celles de JQuery et de DOM, il est donc simple de sélectionner les éléments voulus. Cette librairie nous permettra de mettre en place une solution efficace et optimisée.
En ce qui concerne le parsing Wikitext, nous avons décidé d'utiliser la librairie Sweble Wikitext Parser car c'est celle qui semble la plus sérieuse et utilisable en Java. Cependant, cette librairie ne semble pas couvrir 100% des cas existants de Wikitext. C'est pourquoi nous considérons l'implémentation de l'extracteur Wikitext risqué car n'ayant pu tester la librairie que sur quelques pages à cause du temps d'appropriation.

1.4. Critères d'acceptabilité du produit

L'acceptation du projet sera faite grâce à un concours. Son objectif sera d'extraire le plus possible de tableaux Wikipedia. Et de sortir un jeu de données (un ensemble d'URLs WIkipedia). L'extraction produira bien entendu des fichiers CSV.
Chaque fichier CSV devra contenir les informations d'un seul tableau. Si une page contient plusieurs tableaux, notre programme devra générer plusieurs fichiers CSV.

2. Expression des besoins

2.1. Besoins fonctionnels

- La fonction principale de notre programme sera d'analyser le contenu des différentes pages Wikipedia. Un scan en quelque sorte. Il aura pour but de savoir si une page contient ou non des données tabulaires.
- Dans le cas où des données tabulaires sont présentes sur la page, notre programme devra les extraire et les transformer en tableau CSV.
- Une fois la création des tableaux CSV, l'utilisateur pourra exploiter ces données avec n'importe quel logiciel compatible avec ce format.

2.2. Besoins non fonctionnels

- Notre programme étant codé en java, il devra fonctionner sur n'importe quel système d'exploitation.
- Une contrainte de performance existe. En effet nous sommes mis en concurrence avec les autres groupes de projets. Le but étant d'extraire le plus de tableaux CSV possible dans un grand nombre de page Wikipedia. Il faudra donc avoir le programme java le plus performant possible.
- Une documentation (README.md) du projet en anglais décrivant l'objectif, le résultat, la licence, les technologies utilisés, ainsi que l'architectures du projet sera fourni. Nous devons aussi inclure les instructions pour le déploiement du projet final dans ce document.

3. Contraintes

3.1. Délais

Comme tout projet de cette envergure, celui-ci se doit de respecter un certain nombre de contraintes et celles auxquelles on pense tout d'abord sont d'abord celles du temps. En effet, ce projet se doit de respecter certaines deadlines (dates limites), qui concernent différents rendus du projet concernant différentes étapes de ce dernier.
En premier lieu, il y a la date du 27 octobre 2018, date à laquelle il va falloir rendre un « cahier des charges », document synthétisant les demandes et objectifs du projet ainsi que les outils et prévisions pour y parvenir.
Ensuite viens la date du 20 décembre, correspondant au rendu final du projet, sachant qu'entre temps nous aurons mis en place plusieurs « sprints » que nous allons respecter, qui correspondent à des périodes de temps où différentes tâches sont à effectuer (voir diagramme de Gantt).
Enfin on retrouve la date pour la présentation orale devant un jury se tiendra dans le mois de janvier. Il s'agit d'une présentation du projet pendant 30 minutes au total, rappelant le contexte, présentant le projet et cela à l'aide d'une démonstration visuelle.
Le non-respect de ces délais entraînerait bien évidemment la moins bonne note possible et un refus de l'acceptation du projet.

3.2. Autres contraintes

Ce projet, en plus des contraintes de temps, ne gérant pas les contraintes de coût, se verra ajouter quelques contraintes spécifiques à ce projet. Hormis les contraintes de respect de l'utilisation de certains outils (GitHub), plusieurs autres plus techniques vont venir ajouter de la difficulté sur ce projet.

Notamment, concernant l'analyse des pages Wikipédia, il faudra respecter deux manières possibles de le faire :

- L'une en allant chercher le code Wikitext correspondant
- L'autre en exploitant le rendu HTML de la page Wikipédia

Enfin au niveau des extractions, cela est recommandé et exigé de le faire via HTML et Wikitext, avec une comparaison et des tests de ces deux méthodes. Pour enfin justifier laquelle est meilleure que l'autre.

4. Risques

4.1. Ce qui peut se passer ?

Les projets informatiques sont sujets à plusieurs risques. Quelques-uns peuvent s'appliquer à notre projet.

- Le risque lié au coût du projet est nul car nous n'avons pas de dépense à prévoir pour mener à bien notre projet.
- Le risque lié aux délais du projet est quasiment nul car la date de rendu est fixe et ne bougera vraisemblablement pas.
- Le risque lié aux fonctionnalités est lui bien présent. En effet, nous pouvons oublier ou négliger une fonction au détriment d'autres et donc compromettre le fonctionnement final du projet.
- Les risques quant à la qualité de notre projet sont réels. Il devra, être facile à mettre en place, avoir un temps de réponse faible par rapport à la quantité de données à traiter. De plus les incidents devront être limités de sorte que l'utilisateur utilise du mieux possible notre projet.
- Le risque lié à la facilité d'utilisation de notre projet est réel. L'ergonomie étant un point important dans un projet. Il faudra prendre en compte avec beaucoup de considération cet aspect.
- Nous n'avons ici peu de risques juridiques. En effet, notre projet est voué à une utilisation en interne. Nous prélevons des données de Wikipédia qui est une encyclopédie libre. Il n'y a donc pas de problème de droits.
- Nous pouvons aussi constater un possible problème au moment de l'implémentation de Sweble Wikitext Parser. En effet l'implémentation de la librairie pour l'extraction du code Wikitext n'est pas couverte à 100%. Cela pourrait compromettre la fluidité du programme. De plus nous n'avons tester cette librairie que sur quelque pages en raison du temps d'approximation.

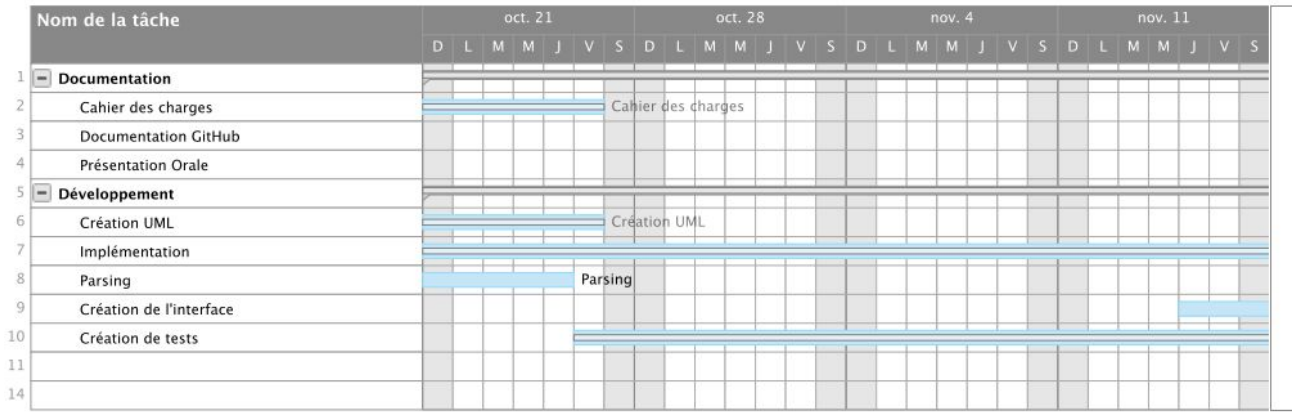
4.2. Traitements possibles

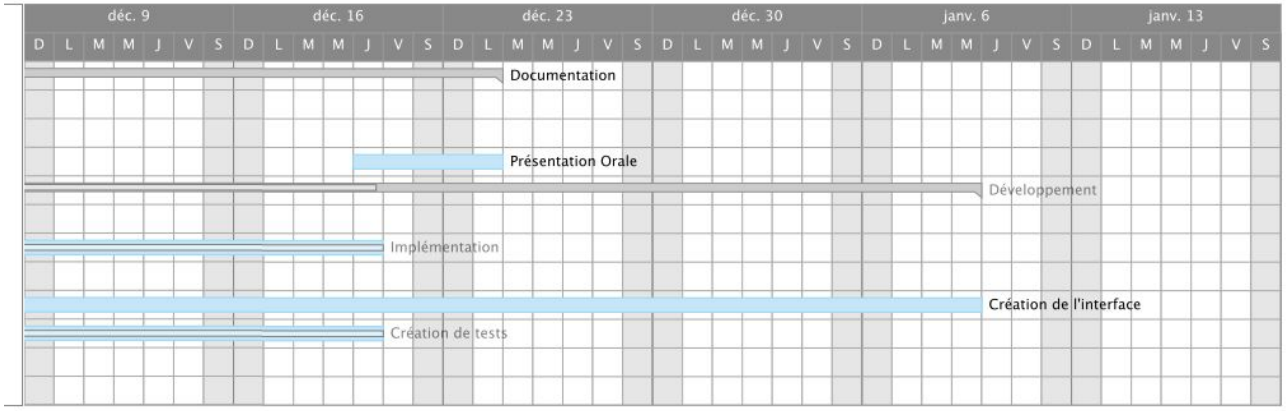
- La gestion du risque lié aux fonctionnalités pourra être traité en effectuant une planification efficace. Et donc un diagramme de Gantt et un suivi de projet efficient. De plus la phase de test du projet peut être un moyen, certe un peu tardif, de se rendre compte qu'une fonctionnalité n'est pas implémenter correctement ou purement oublié.
- La gestion du risque lié à la qualité de notre projet sera importante. La principale méthode de gestion de cette qualité réside dans la phase de test de notre projet. Par ailleurs, une optimisation du code pourra être fait en amont, au moment de la programmation. Ce travail sera une des tâches les plus importantes du projet.

- La gestion de la facilité d'utilisation du projet sera réalisé avant la phase de programmation. Cette phase réalisé en amont pourra consister en une phase de maquettage de l'interface utilisateur. Cette phase aura deux objectifs. Elle nous -l'équipe de projet- permettra de nous mettre d'accord sur le comportement général du projet et donc d'avancer dans la même direction tout au long du projet. Elle permettra par ailleurs d'évaluer la faisabilité du projet tel que nous le percevons et de changer en conséquence son comportement.
- Concernant l'extension Sweble Wikitext Parser, nous devons l'utiliser avec une quantité de pages croissante. En commençant par un petit nombre de pages. Cela nous donnera une idée plus globale du fonctionnement de cette extension. Nous pourrons ensuite tirer plus facilement profit de sa puissance.

5. Déroulement du projet

5.1. Planification



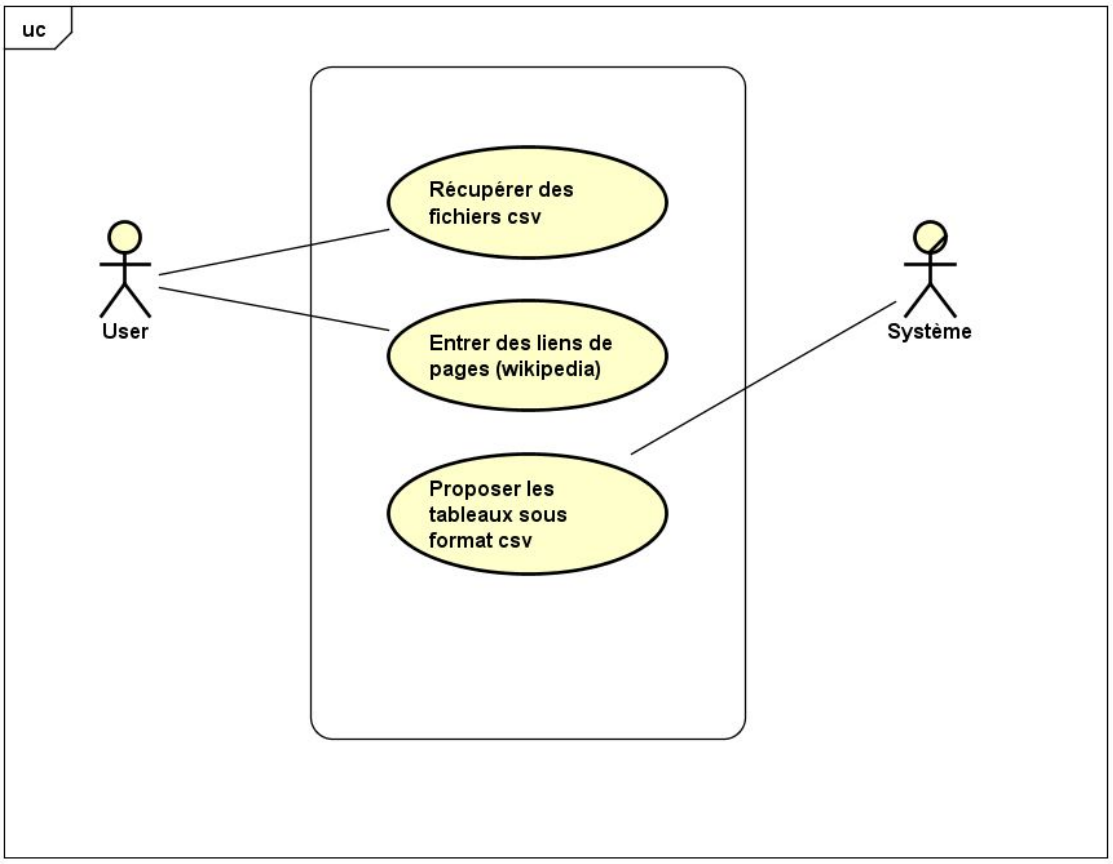


Note : des semaines ne sont pas affichées entre les deux parties du diagramme car aucune tâche ne commence ou ne se termine.

5.2. Documentation

5.2.1. UML

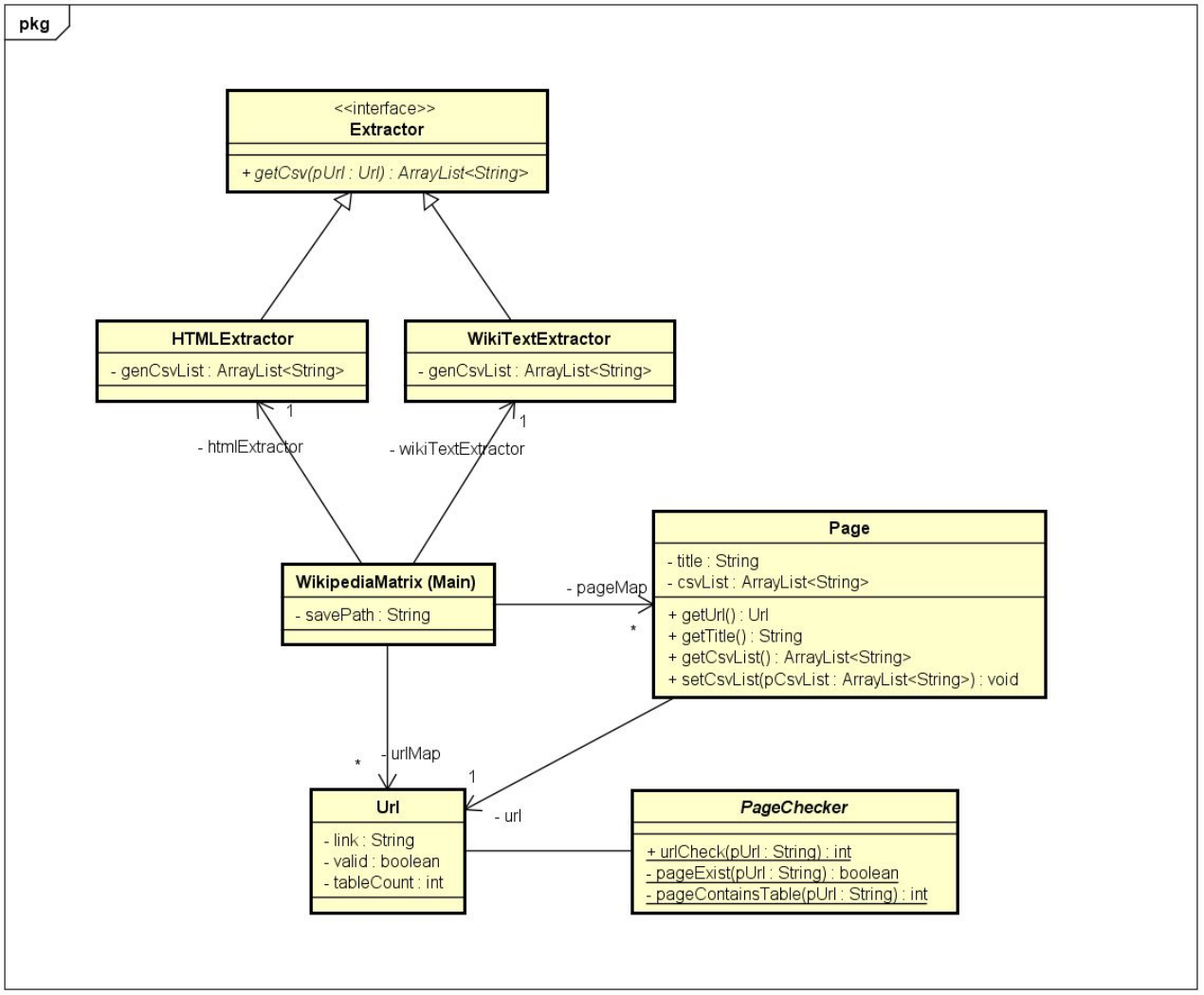
1) <u>Le diagramme de cas d'utilisation</u>



L'objectif de l'application étant de proposer à l'utilisateur les tableaux présents sur les pages Wikipédia dont il fournira les liens, ce dernier n'a que 2 interactions possibles avec l'application.

Le système répondra à l'utilisateur en lui proposant les tableaux au format CSV qu'il pourra enregistrer.

2) Le diagramme de classes



L'extraction des tableaux devant être possible depuis les formats HTML et Wikitext, des technologies différentes sont alors employées, d'où la nécessité de séparer les deux approches et de spécifier ce que les deux extracteurs doivent implémenter.

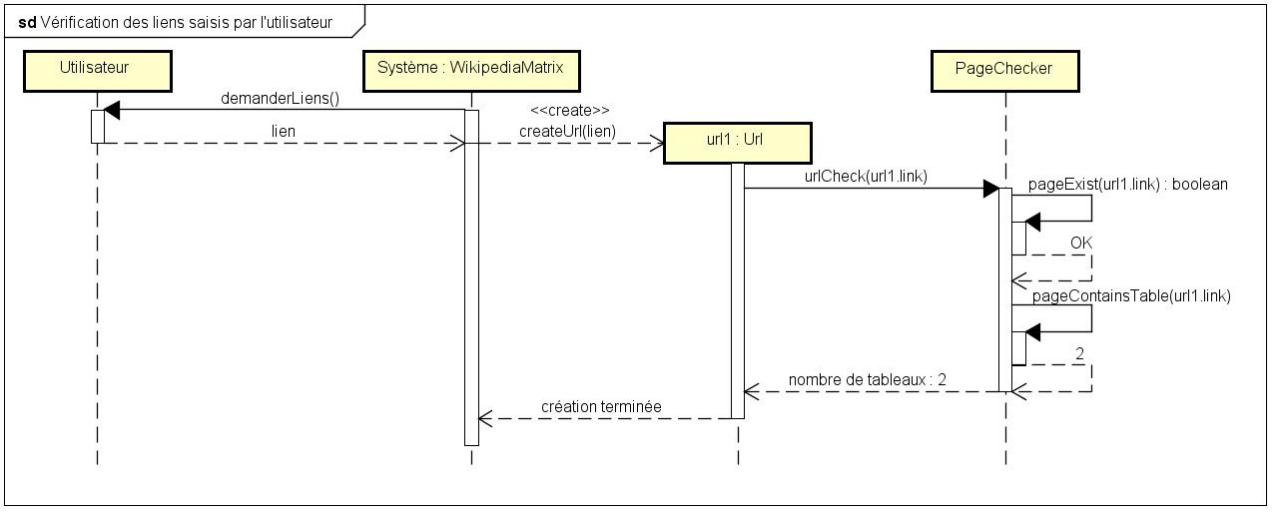
La classe WikipediaMatrix orchestre les étapes menant à la génération des fichiers CSV attendus.

La classe Url contient différentes informations comme le lien entré par l'utilisateur, un attribut désignant la page comme exploitable ou non et le nombre de tableau(x) présent(s) sur la page.

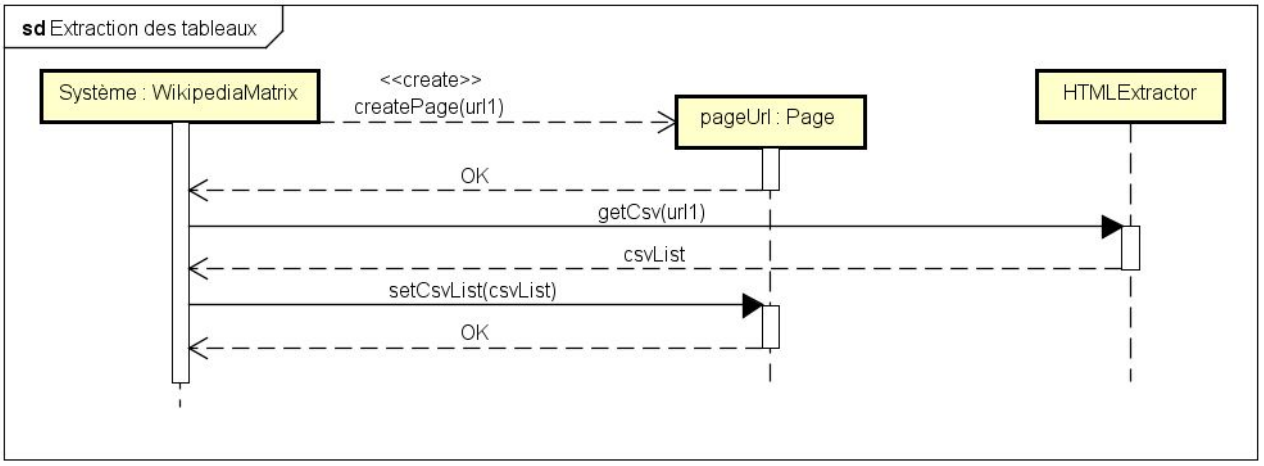
La classe PageChecker agit tel un filtre. Cette classe vérifie tous les liens saisis par l'utilisateur afin de désigner ceux exploitables par les extracteurs.

La classe Page contient l'url vérifiée et la liste des CSV des tableaux extraits. Elle est utilisée en tant que conteneur des informations intéressantes de la page, ici le titre et les tableaux.

3) Les diagrammes de séquence



Ici une procédure standard de saisie de liens par l'utilisateur est représentée, les liens sont d'abord utilisés pour créer un objet qui sera ensuite complété par les retours des vérifications effectuées sur le lien. Par exemple, dans le cas où la le lien ne provient pas de Wikipedia, PageChecker n'effectuera pas le comptage du nombre de tableaux et retournera "-1" à l'objet Url qui mettra sa validité à "faux" et son nombre de tableaux à "0". Les objets Url "invalides" ne seront pas utilisés dans la suite de la procédure.



Dans le cas où des liens valides existent, le programme créera les objets Page correspondants et fera appel à un des extracteurs afin de récupérer les tableaux et les ajouter à leur objet Page correspondant.

5.2.2. Scénario

Si nous n'avons pas les moyens de réaliser l'interface graphique de l'application, son utilisation standard sera la suivante :

1. Au lancement de l'application, l'utilisateur devra entrer des liens Wikipedia un par un et sortira de la saisie en entrant la valeur spécifiée..
2. L'application entrera ensuite dans une phase de vérification de tous les liens ajoutés par l'utilisateur.
3. Si au moins un lien est considéré valide, l'application procédera à la création des objets nécessaires et passera alors à l'étape d'extraction.
4. L'étape d'extraction est l'étape qui va récupérer les données des tableaux et qui va également créer les fichiers CSV correspondants.
5. Une fois l'extraction terminée, l'utilisateur pourra télécharger les fichiers CSV générés dans le dossier de son choix.

Si l'interface graphique est créée, son utilisation sera radicalement identique mais des aspects visuels permettront de connaître des informations supplémentaires comme la validité d'un lien et son nombre de tableaux disponibles.