RAPPORT DE PROJET : *Web Api*

Le but du projet est d'utiliser la librairie python ***bottle***, pour ré-implémenter l'API du site ***dbpl*** qui regroupe l'ensemble des publications en informatique. Ce rapport explique les différentes routes ainsi que les fonctions utilisées dans le code.

1. Parser le fichier xml

La première phase du projet a été de parser le fichier xml afin de créer une structure assez simple à manipuler. Il est vrai que python propose plusieurs librairies pour parser un fichier xml mais dans le cadre de ce projet j'ai utilisé la librairie python ''Sax'' qui est assez rapide et permet de traiter une quantité très importante de données. Le parseur parcours le fichier, recherche les publications de type : article, inproceedings, proceedings, book, incollection, phdthesis, ou masterthesis. Il récupère ensuite les champs : year, tiltle, journal et booktitle de ces différentes publications et créé une liste de dictionnaires contenant toutes ses informations. Cette liste est globale et sera utilisée par toutes les routes et fonctions de l'application.

2. Route *: /publications/{id}*

Cette première route est assez simple, elle permet d'obtenir toutes les informations sur la publication à la position **id** dans la liste. Elle vérifie que l'id est compris entre 0 et la taille de la liste, si oui alors elle renvoie le dictionnaire en format json, sinon elle renvoie un message d'erreur.

3. Route *: /authors/<name>*

Le paramètre ***name*** est le nom complet d'un auteur. Si l'auteur existe alors la route retourne le nombre de publications que ce dernier a publié ainsi que le nombre de co-auteurs avec lesquels il a travaillé. Pour cela il a fallu implémenter deux fonctions distinctes :

- ***listePublications*** : prend en paramètre le nom d'un auteur et retourne la liste de ses publications.
- ***listeCoAuteur*** : prend en paramètre le nom d'un auteur et retourne la liste de co-auteurs avec lesquels il a publié.

Ensuite la route retourne la taille de ces différentes listes.

4. Route *: /authors/<name>/publications*

Cette route retourne la liste des publications d'un auteur au format json. Elle utilise la fonction ***listePublications***, présentée un peu plus haut.

5. Route : */authors/<name>/coauthors*

Cette route retourne la liste des co-auteurs d'un auteur au format json. Elle utilise la fonction ***listeCoAuteur***, présentée un peu plus haut.

6. Route: */search/authors /<SearchString>*

Cette route permet de faire une recherche sur les auteurs. Cette recherche est basée sur le parametre ''searchstring'' fourni par l'utilisateur. On peut intégrer des caractères spéciaux tels que ''* '' pour dire ''un ou plusieurs caractères '' et ''%'' pour dire ''un caratère quelconque. La route retourne, au format json, 1 à plusieurs noms d'auteurs répondant au paramètre si il en existe.

7. Route: */search/publications /<SearchString>*

Cette route fait la même chose que la précédente, seulement la recherche se fait sur les titres des publications et non sur les auteurs. Elle retourne donc toutes les publications répondant au paramètre ''searchstring''.

8. Route: */authors /<name_origine>/distance/<name_destination>*

Cette route permet de calculer le plus court chemin entre deux auteurs : name_origine et name_destination. Si les deux auteurs ont travaillé, ensemble, sur une publication alors la valeur du chemin est 0. Dans le cas contraire, l'algorithme cherche un lien entre ces deux auteurs et la valeur du chemin correspond au nombre de sauts c'est-à-dire le nombre d'auteurs intermédiaires. Cette route utilise l'algorithme de DIJKSTRA pour trouver ce chemin. Une première phase est de construire un arbre des auteurs qui est ensuite fourni comme paramètre pour la fonction. Cet algorithme est assez performant dans le sens où il ne recherche pas tous les chemins avant de choisir le plus court mais il construit au fur et à mesure le plus court chemin. Cette fonction est récursive.

9. Fonction de découpage : *FonctDecoupe()*

Cette fonction permet de prendre en charge les différentes fonctionnalités de l'application : le nombre d'enregistrement, les champs à afficher, le tri des résultats suivant une colonne. Elle est appelée dans toutes les routes qui retournent une liste de résultats ; en fonction des paramètres de l'utilisateur elle effectue les traitements adaptés pour retourner répondant aux critères.