PROGRES – Mini-Projet 2

MERLIN Lucas
28705933
lucas.merlin@etu.sorbonne-universite.fr

Pour réaliser ce premier mini-projet, j'ai travaillé en autonomie.
Ce mini-projet avait pour but de se servir de la bibliothèque bottle proposé en Python pour réaliser une API au site http://dblp.uni-trier.de/db/ht qui regroupe l'ensemble des publications scientifiques en informatique puis un site web afin d'utiliser cette API.

Durant la réalisation de ce projet, j'ai utilisé les données du fichier « dblp_2019_2020.xml » afin de pouvoir l'exécuter sur ma machine personnelle qui ne possède que très peu de RAM. Cependant, nous pouvons utiliser le travail effectué sur un plus petit ou bien un plus grand nombre de publications en modifiant simplement la variable « local_input » au tout début de api.py en y mettant le fichier sur lequel nous voulons travailler.
Pour commencer, j'ai donc parser le fichier grâce à lxml pour pouvoir parcourir les différentes publications plus facilement.

Par la suite, j'ai donc commencé à implémenter l'API web sur le port 8080.

La première fonctionnalité de l'API que j'ai implémenté est pour l'URL « *publications*/{id} » qui permet d'afficher toutes les informations trouvées pour la publication « id ». Pour cela, on parcourt l'intégralité des balises <title> et compare si cela correspond a « id ». SI tel est le cas, on récupère toutes les autres balises liées à celle-ci puis les affiches. c

La deuxième est pour l'URL « /publications ». Cet URL retourne par défaut les 100 premières publications en parcourant l'arbre et en stockant les informations de chaque publications dans une liste. Cette fonctionnalité peut prendre en paramètre la valeur x pour :

- start : saute les x premières publications avant de commencer à lister.
- count/limit : les deux paramètres ont ici la même utilité qui est d'afficher x publications.
- order : trie les publications de manière lexicographique avant de les afficher en fonction de la balise x.

Ce troisième URL « *authors*/{name} » à pour but de retourner le nombre de publications auxquelles l'auteur « name » a participé ainsi que le nombre de co-auteurs avec lesquelles il a collaboré. Ces valeurs sont retournés en parcourant les balises<author> et si elle correspond à « name », nous incrémentons le nombre de publications ainsi que le nombre de co-auteurs en fonction du nombre présent sur celle ci. Cette fonctionnalité peut prendre en paramètre la valeur x pour :

- start : saute les x premiers auteurs avant de commencer à incrémenter.
- count: affiche le nombre de co-auteurs pour x publications.

Si l'auteur n'est pas trouvé, on retourne une erreur 404.

L’URL suivant est « /authors/{name}/publications ». Il permet de retourner la liste des publications auxquelles l’auteur « name » a participé. On effectue cela en parcourant les balises <author> en les comparants avec « name », si il y a correspondance, on liste la publications concernée. Cette fonctionnalité peut prendre en paramètre la valeur x pour :

- start : saute les x premières publications avant de commencer à lister.
- count: affiche les x premières publications.
- order : trie les publications de manière lexicographique avant de les afficher en fonction de la balise x.

Si l’auteur n’est pas trouvé, on retourne une erreur 404.

« /authors/{name}/coauthors » est très similaire à la fonctionnalité précédente car à la place de lister la publication liée à l’auteur, on liste les auteurs présents sur cette publication tout en veillant que nous avons pas déjà stocké cet auteur afin d’éviter les doublons. Cette fonctionnalité peut prendre en paramètre la valeur x pour :

- start : saute les x premiers auteurs avant de commencer à lister.
- count: affiche les x premiers co-auteurs.
- order : trie les co-auteurs de manière lexicographique si et seulement si x vaut « author ».

Si l’auteur n’est pas trouvé, on retourne une erreur 404.

La sixième fonctionnalité est liée à l’URL « /search/authors/{searchString} ». Elle retourne la liste des auteurs qui ont dans leur nom la chaîne « searchString ». On parcourt donc les balises <author> et on vérifie si la chaîne y est présente, si oui on liste l’auteur. Cette fonctionnalité peut prendre en paramètre la valeur x pour :

- start : saute les x premiers auteurs avant de commencer à lister.
- count: affiche les x premiers auteurs.
- order : trie les auteurs de manière lexicographique si et seulement si x vaut « author ».

Si l’auteur n’est pas trouvé, on retourne une erreur 404.

Encore une fois, l’URL « /search/publications/{searchString} » liste de manière similaire a l’URL précédent les publications qui contiennent la chaîne « searchString » dans leur titre. On parcourt alors cette fois ci les balises <title> et stocke l’intégralité des informations de la publications si il y a une similitude. Cette fonctionnalité peut prendre en paramètre la valeur x pour :

- start : saute les x premières publications avant de commencer à lister.
- count: affiche les x premières publications.
- order : trie les publications de manière lexicographique avant de les afficher en fonction de la balise x.
- balise : ce paramètre prend en nom une balise y et en valeur une chaîne de caractère x, les publications alors stockés sont testés pour vérifier si leur balise y contienne la chaîne x. Si non, la publications est retiré de la liste.

Si la publication n’est pas trouvé, on retourne une erreur 404.

La dernière fonctionnalité implémentée est l’URL « /authors/{name_origin}/distance/{name_destination} » qui part de l’auteur « name_origin » et on parcourt l’intégralité des co-auteurs de manière récursive en stockant le chemin qu’on parcourt afin de trouver l’auteur « name_destination ». On retourne alors le plus petit chemin qu’on trouve ainsi que la distance séparant les deux auteurs.

Si aucun chemin n’est trouvé, on retourne une erreur 404.

J’ai par la suite réalisé les tests unitaires pour l’API web.
Pour cela, j’ai donc utilisé unittest est réalisé pour chaque fonctionnalité de l’API différents tests. Les tests couvrent les paramètres pour chaque fonctionnalités afin de s’assurer de leur bon fonctionnement.

Pour finir, on peut accéder à une interface web grâce a bottle sur le port 8081.

Une première interface propose la saisie d’un nom d’auteur afin d’obtenir l’intégralité de ses publications ainsi que les co-auteurs avec qui il a collaboré.
Pour y accéder, l’URL est « *authors/*name/ » qui envoie une requête POST et récupère les données proposées par l’API web aux URLs « /authors/{name}/coauthors » et
« /authors/{name}/publications » afin de les affichés en retour sur l’interface web.

La seconde interface propose d’insérer deux noms d’auteurs et d’avoir la distance qui les sépare ainsi que le chemin à l’URL "/authors/distance". Cette URL envoie une requête post et récupère les données proposées par l’API web à l’URL « /authors/{name1}/distance/{name2} » et les affiches sur l’interface web.

Ce mini-projet m’a donc permit de mieux comprendre et implémenter une API web.